1. Company Name

Smile Dentistry

2. Project Title

Dental Clinic Appointment Management Database

3. Team

Stanley Chan (MySQL Expert)
Markus Chow (MS SQL Server Expert)

4. Weekly Meeting Hours

We will meet every Tuesday from 10:30am – 11:30am. If needed, the meeting can be extended to 12:30, or an extra meeting can be scheduled on Thursday (10:30am – 11:30am) in the same week.

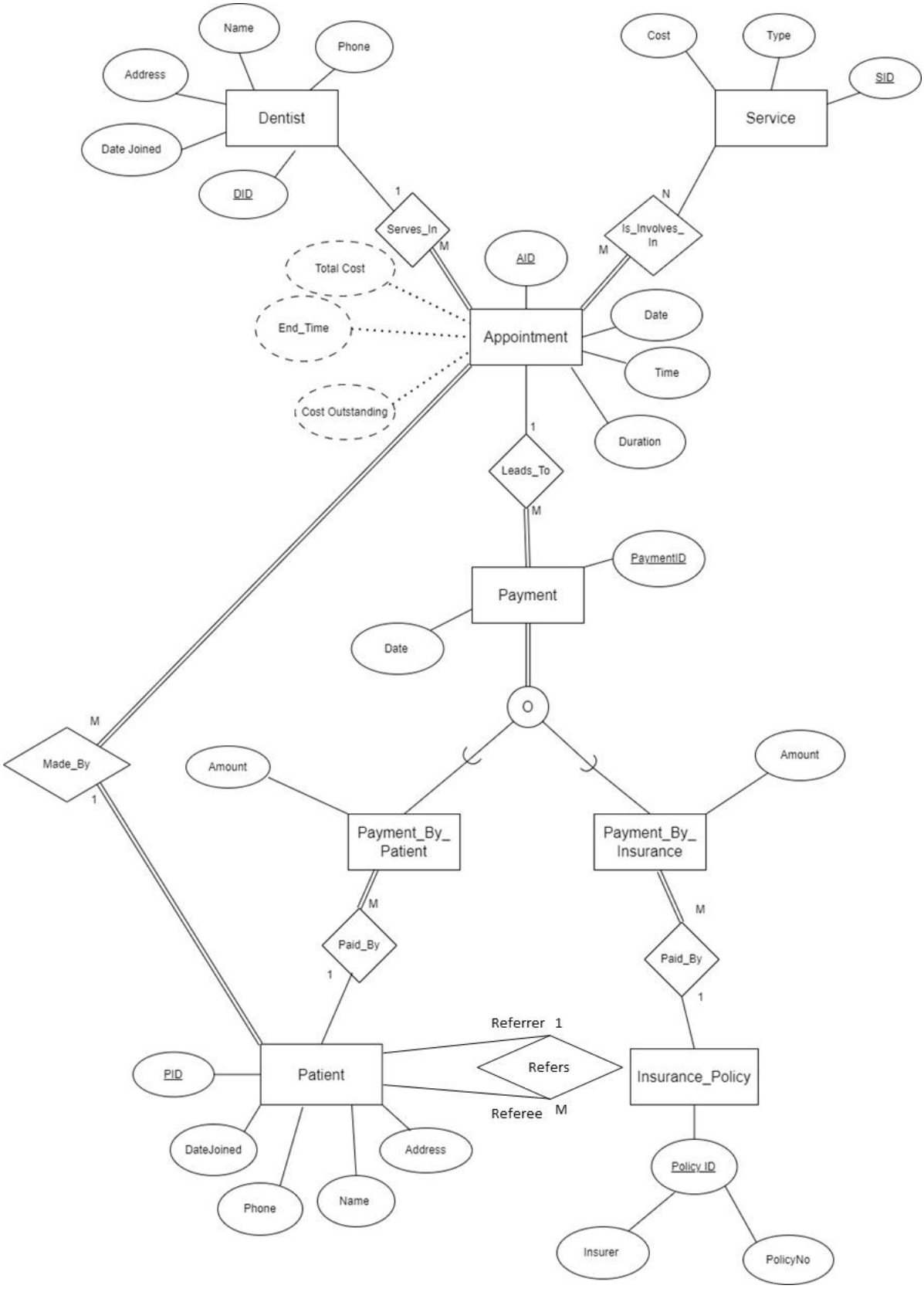
5. Project Description

There is a dental clinic managed by several dentists. Each dentist receives some patients. For each patient, several appointments are booked on different days and times. Each patient may have one or more insurance policies, which can cover a payment fully or partially. It is assumed that, in most payments, a patient pays partially with one or more insurance policies. We need to store the information of insurance policies of the patients. Each insurance policy under the same insurer has its own unique policy number, but the same number may be used by different insurance companies. Some patients may not have an insurance policy and they pay the expenses from their own pocket. We also need to store the payment history of the patients. A patient may pay all expenses at once or in different installments. Patients can refer to their friends and families and we store this information in the database too.

6. Assumptions about Cardinality and Participations

- A dentist may have no patient (e.g. when a new dentist joins the clinic).
- A person can only be registered as a patient after making the very first appointment.
- A patient may refer zero or more referees (who are also patients).
- A patient can only have at most one referrer (who is also a patient).
- Each appointment is made between one patient and one dentist.
- One or more services can be provided in one appointment.

7. EER Diagram



8. ER-Model Mapping to Database Relational Schema[1]

Dentist(DID, Name, Phone, Address, DateJoined)
Service(SID, Type, Cost)
Patient(PID, Name, Phone, Address, DateJoined)
Appointment(AID, Date, Time, Duration, **DID**, **PID**)
Insurance_Policy(Insurer, PolicyNo)
Payment(PaymentID, Date, isPaidByPatient, isPaidByInsurance, PatientAmount, PolicyAmount, **AID**, **PID**, **(Insurer, PolicyNo)**)
Service_Involved_In_Appointment(**SID, AID**)
Referral(**RefereePID, ReferrerPID**)

9. Normalization

First normal form
In 1NF, there should be:

- No composite attribute
- No multivalued attribute
- No nested table

All of them are absent in the current relations. Therefore, the schema is in 1NF.

Second normal form
In 2NF, there should be no partial (key) functional dependency (FD).
Except for Service_Involved_In_Appointment, all relations have single primary key. Therefore, there is no partial (key) FD in them.
As for Service_Involved_In_Appointment, there are no non-prime attributes. Therefore, there is no partial (key) FD in it.
Therefore, the schema is in 2NF.

Third normal form
In 3NF, there should be no transitive FD.
In Payment relation:

- PaymentID → AID
- AID → PID (Because each appointment must be made by one particular patient.)
- Meanwhile, AID is not a candidate key in Payment relation because each appointment can lead to multiple payments with different date, amount, etc.
- Therefore, PaymentID → AID → PID is a transitive FD.

[1] Steps of mapping are attached in Appendix 1 for reference.

Without any decomposition, there is insertion anomaly when entering data into the Payment table – in each row, the AID and PID must be corresponding to each other, meaning that there is a waste of effort and a greater chance of making mistakes.

To decompose Payment relation, the following relations are resulted:

- Payment(PaymentID, Date, isPaidByPatient, isPaidByInsurance, PatientAmount, PolicyAmount, **AID**, **(Insurer, PolicyNo)**)
- Appointment_Patient(**AID**, **PID**)

However, the Appointment_Patient relation can be ignored because the original Appointment relation can already serve the purpose. Therefore, to achieve 3NF, attribute PID is removed from the Payment relation.

Below is the collection of relations in 3NF:

- Dentist(DID, Name, Phone, Address, DateJoined)
- Service(SID, Type, Cost)
- Patient(PID, Name, Phone, Address, DateJoined)
- Appointment(AID, Date, Time, Duration, **DID**, **PID**)
- Insurance_Policy(Insurer, PolicyNo)
- Payment(PaymentID, Date, isPaidByPatient, isPaidByInsurance, PatientAmount, PolicyAmount, **AID**, **(Insurer, PolicyNo)**)
- Service_Involved_In_Appointment(**SID, AID**)
- Referral(**RefereePID**, **ReferrerPID**)

BCNF

In BCNF, in each of the FDs listed, the LHS attribute must be a superkey.

Functional dependency	Evaluation
Dentist(DID, Name, Phone, Address, DateJoined)	
DID → Name	OK because DID is the primary key.
DID → Phone	
DID → Address	
DID → DateJoined	
Phone → DID	OK because Phone should be a superkey (if we assume that each phone number is only owned by one dentist).
Phone → Name	
Phone → Address	
Phone → DateJoined	
Service(SID, Type, Cost)	
SID → Type	OK because SID is the primary key.
SID → Cost	
Type → SID	OK because Type should be a superkey (if we assume that the name of each service type is unique).
Type → Cost	

Functional dependency	Evaluation
Patient(PID, Name, Phone, Address, DateJoined)	
PID → Name	OK because PID is the primary key.
PID → Phone	
PID → Address	
PID → DateJoined	
Phone → PID	OK because Phone should be a superkey (if we assume that each phone number is only owned by one patient).
Phone → Name	
Phone → Address	
Phone → DateJoined	
Appointment(AID, Date, Time, Duration, **DID**, **PID**)	
AID → Date	OK because AID is the primary key.
AID → Time	
AID → Duration	
AID → DID	
AID → PID	
{Date, Time, PID} → AID	OK because {Date, Time, PID} and {Date, Time, DID} are both superkeys – each patient / dentist can only attend one appointment at one particular time!
{Date, Time, PID} → Date	
{Date, Time, PID} → Time	
{Date, Time, PID} → Duration	
{Date, Time, PID} → DID	
{Date, Time, DID } → AID	
{Date, Time, DID } → Date	
{Date, Time, DID } → Time	
{Date, Time, DID } → Duration	
{Date, Time, DID } → PID	
Insurance_Policy(Insurer, PolicyNo)	
No functional dependency.	
Payment(PaymentID, Date, isPaidByPatient, isPaidByInsurance, PatientAmount, PolicyAmount, **AID**, **(Insurer, PolicyNo)**)	
PaymentID → Date	OK because PaymentID is the primary key.
PaymentID → isPaidByPatient	
PaymentID → isPaidByInsurance	
PaymentID → PatientAmount	
PaymentID → PolicyAmount	
PaymentID → AID	
PaymentID → {Insurer, PolicyNo}	

Functional dependency	Evaluation
Service_Involved_In_Appointment(**SID, AID**)	
No functional dependency.	
Referral(**RefereePID, ReferrerPID**)	
No functional dependency.	

After checking, the current relations are in BCNF.

Result

To sum up, after normalization, the relations are as follows:
Dentist(DID, Name, Phone, Address, DateJoined)
Service(SID, Type, Cost)
Patient(PID, Name, Phone, Address, DateJoined)
Appointment(AID, Date, Time, Duration, **DID**, **PID**)
Insurance_Policy(Insurer, PolicyNo)
Payment(PaymentID, Date, isPaidByPatient, isPaidByInsurance, PatientAmount, PolicyAmount, **AID**, **(Insurer, PolicyNo)**)
Service_Involved_In_Appointment(**SID, AID**)
Referral(**RefereePID, ReferrerPID**)

10. Determining Data Types (Domain) and Constraints

Dentist(DID, Name, Phone, Address, DateJoined)

Field	Data type	Other constraints
DID	INT ● For storing a possibly large amount of dentists	PRIMARY KEY ● For uniquely identifying each dentist
Name	VARCHAR(100) ● Variable length of string to store names with different number of characters ● Large limit to allow possibly long names from certain cultures	NOT NULL ● Must enter a name for the patients and staff to identify the dentist
Phone	CHAR(10) ● Fixed length to store a presumably Canadian local phone number	NOT NULL ● Must enter a phone number for easy contact in urgent cases
Address	VARCHAR(200) ● Variable length of string to store addresses with different number of characters ● Large limit to allow possibly long addresses	NOT NULL ● Must enter an address for delivering documents such as tax slips
DateJoined	DATE	

Service(SID, Type, Cost)

Field	Data type	Other constraints
SID	INT ● For storing a possibly large amount of services	PRIMARY KEY ● For uniquely identifying each service
Name	VARCHAR(100) ● Variable length of string to store services with different number of characters ● Large limit to allow greater flexibility	NOT NULL ● Must enter a name staff to identify the services more easily
Cost	NUMERIC(10, 2) ● Large precision to allow greater flexibility to store more expensive services ● Scale fixed to 2 for storing the cost rounded to cent	NOT NULL ● Nothing comes without a cost!

Patient(PID, Name, Phone, Address, DateJoined)

Field	Data type	Other constraints
PID	INT ● For storing a possibly large amount of dentists	PRIMARY KEY ● For uniquely identifying each patient
Name	VARCHAR(100) ● Variable length of string to store names with different number of characters ● Large limit to allow possibly long names from certain cultures	NOT NULL ● Must enter a name for the staff to identify the patient
Phone	CHAR(10) ● Fixed length to store a presumably Canadian local phone number	NOT NULL ● Must enter a phone number for easy contact in urgent cases
Address	VARCHAR(200) ● Variable length of string to store addresses with different number of characters ● Large limit to allow possibly long addresses	NOT NULL ● Must enter an address for delivering documents such as checkup reports
DateJoined	DATE	

Appointment(AID, Date, Time, Duration, **DID**, **PID**)

Field	Data type	Other constraints
AID	INT ● For storing a possibly large amount of appointments	PRIMARY KEY ● For uniquely identifying each appointment
Date	DATE	NOT NULL ● Must enter for staff's reference
Time	TIME	NOT NULL ● Must enter for staff's reference
Duration	TIME	NOT NULL ● Must enter for staff's reference ● To use ADDTIME() to derive the end time of the appointment[2]
DID	INT ● Same as DID of Dentist table	FOREIGN KEY ● Referencing DID of Dentist table NOT NULL ● Must enter to refer to a particular dentist
PID	INT ● Same as PID of Patient table	FOREIGN KEY ● Referencing PID of Patient table NOT NULL ● Must enter to refer to a particular patient

Insurance_Policy(Insurer, PolicyNo)

Field	Data type	Other constraints
Insurer	VARCHAR(30) ● For storing the name of the insurer	PRIMARY KEY ● Composite primary key to unique identify each insurance policy
PolicyNo	VARCHAR(30) ● For storing the number of the insurance policy ● Not using a purely numeric data type to cater policy numbers with letter(s) used by some insurers	

[2] https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html#function_addtime

Payment(PaymentID, Date, isPaidByPatient, isPaidByInsurance, PatientAmount, PolicyAmount, **AID**, **(Insurer, PolicyNo)**)

Field	Data type	Other constraints
PaymentID	INT ● For storing a possibly large amount of payments ● No negative value allowed	PRIMARY KEY ● For uniquely identifying each payment
Date	DATE	NOT NULL ● Must enter for staff's reference
isPaidByPatient	TINYINT ● Used to handle Boolean data type, which is not available in MySQL[3]	NOT NULL ● Must enter for staff's reference CHECK (isPaidByPatient >= 0 AND isPaidByPatient <= 1) ● Ensure that the value is either 0 (false) or 1 (true)
isPaidByInsurance		NOT NULL ● Must enter for staff's reference CHECK (isPaidByPatient >= 0 AND isPaidByPatient <= 1) ● Ensure that the value is either 0 (false) or 1 (true)
PatientAmount	NUMERIC(10, 2) ● Same as the Cost field of Service table ● Large precision to allow greater flexibility to store more expensive services ● Scale fixed to 2 for storing the cost rounded to cent	
PolicyAmount		
AID	INT ● Same as AID of Appointment table	FOREIGN KEY ● Referencing AID of Appointment table NOT NULL ● Must enter to refer to a particular appointment
Insurer	VARCHAR(30) ● Same as Insurer of Insurance_Policy table	FOREIGN KEY ● Composite foreign key referencing {Insurer, PolicyNo} of Insurance_Policy table
PolicyNo	VARCHAR(30) ● Same as PolicyNo of Insurance_Policy table	

[3] https://dev.mysql.com/doc/refman/8.0/en/other-vendor-data-types.html. Not including any width for TINYINT because of warning 1681 (Integer display width is deprecated and will be removed in a future release) as mentioned in MySQL.

Service_Involved_In_Appointment(**SID, AID**)

Field	Data type	Other constraints
SID	INT ● Same as SID of Service table	FOREIGN KEY ● Referencing SID of Service table
AID	INT ● Same as AID of Appointment table	FOREIGN KEY ● Referencing AID of Appointment table

Note: {SID, AID} together as the composite primary key of this table.

Referral(**RefereePID, ReferrerPID**)

Field	Data type	Other constraints
RefereePID	INT ● Same as PID of Patient table	PRIMARY KEY ● For uniquely identifying patient referred by another one FOREIGN KEY ● Referencing PID of Patient table
ReferrerPID	INT ● Same as PID of Patient table	FOREIGN KEY ● Referencing PID of Patient table

11. Creating Database and Tables - SQL DDL

Please see proj_createtables_mysql.sql (for MySQL) and proj_createtables_mssql.sql (for MS SQL Server).

12. Inserting Values in Tables

Please see proj_insertvalues_mysql.sql (for MySQL) and proj_insertvalues_mssql.sql (for MS SQL Server). For sample data used, please see Appendix 2.

13. SQL Queries

Please see proj_queries_mysql.sql (for MySQL) and proj_queries_mssql.sql (for MS SQL Server).

14. Views

There is one view to allow the clinic to see the future appointments more easily. Please see query 4 of proj_queries_mysql.sql (for MySQL) and proj_queries_mssql.sql (for MS SQL Server) for more information.

Appendix 1: Steps of ER-Model Mapping

Step 1: Map strong entities (including specialization)
Dentist(DID, Name, Phone, Address, DateJoined)
Service(SID, Type, Cost)
Appointment(AID, Date, Time, Duration)
Patient(PID, Name, Phone, Address, DateJoined)
Insurance_Policy(Insurer, PolicyNo)
Payment(PaymentID, Date, isPaidByPatient, isPaidByInsurance, PatientAmount, PolicyAmount)
(See Notes about Payment relation below.)

Notes about Payment relation

Option 3: One superclass with one type attribute
Payment(PaymentID, Date, PaymentType, PatientAmount, PolicyAmount)
➔ **Not valid** because of the overlapping constraint

Option 1: Superclass + Subclasses
Payment(PaymentID, Date)
Payment_By_Patient(**PaymentID**, Amount)
Payment_By_Insurance(**PaymentID**, Amount)

Option 2: Just subclasses
Payment_By_Patient(PaymentID, Date, Amount)
Payment_By_Insurance(PaymentID, Date, Amount)

Option 4: One superclass with more than one type attributes
Payment(PaymentID, Date, isPaidByPatient, isPaidByInsurance, PatientAmount, PolicyAmount)

- Options 1, 2, 4 are valid.
- Our current assumption is that (a) payments by ***both*** patient ***and*** insurance is more common than (b) payments by ***either*** patient ***or*** insurance. It means that for each Payment, it is more likely be Paid_By_Patient and Paid_By_Insurance ***at the same time***.
- In that case, Option 1 can be better than Option 2 because redundancy of PaymentID and Date happens more frequently in the latter.
- Because of the assumption above, Option 4 should also lead to not many null values.
- When comparing Options 1 and 4:
    - Option 1 leads to smaller tables and less null values, but more effort to join tables when we want to retrieve the details of each payment.
    - Option 4 leads to more null values and a larger table, but less effort to join tables when we want to retrieve the details of each payment.
- Option 4 is tentatively chosen here.

Step 2: Map weak entities
No weak entity

Step 3: Map 1:1 binary relationship
No 1:1 binary relationship

Step 4: Map 1:M binary relationship
Dentist_Serves_In_Appointment? (1:M)
- Dentist(DID, Name, Phone, Address, DateJoined)
- Appointment(AID, Date, Time, Duration, **DID**)

Appointment_Made_By_Patient? (M:1)
- Appointment(AID, Date, Time, Duration, **DID**, **PID**)
- Patient(PID, Name, Phone, Address, DateJoined)

Appointment_Leads_To_Payment? (1:M)
- Appointment(AID, Date, Time, Duration, **DID**, **PID**)
- Payment(PaymentID, Date, isPaidByPatient, isPaidByInsurance, PatientAmount, PolicyAmount, **AID**)

Payment_Paid_By_Patient? (M:1)
- Payment(PaymentID, Date, isPaidByPatient, isPaidByInsurance, PatientAmount, PolicyAmount, **AID**, **PID**)
- Patient(PID, Name, Phone, Address, DateJoined)

Payment_Paid_By_Insurance? (M:1)
- Payment(PaymentID, Date, isPaidByPatient, isPaidByInsurance, PatientAmount, PolicyAmount, **AID**, **PID**, **(Insurer, PolicyNo)**)
- Insurance_Policy(Insurer, PolicyNo)

Notes about options of mapping the 1:M relationships
In all the 1:M relationship, all the many-side participations are total. Therefore, relationship relations are not used here.

Step 5: Map M:N relationship
Service_Involved_In_Appointment?
- Service_Involved_In_Appointment(**SID, AID**)

Step 6: Map recursive relationship
Referrer_Refers_Referee? (1:M)
- Referral(**RefereePID, ReferrerPID**)

Notes about options of mapping this recursive relationship
In this 1:M recursive relationship, the many-side participations is partial. Therefore, relationship relations is used here to avoid unnecessary null values.

Step 7: Map multivalued attributes
No multivalued attributes

Appendix 2: Sample data in tables

Dentist

DID	Name	Phone	Address	DateJoined
1	Den Smile	1234567890	1234 Smiley Street V6E 3F4	**NULL**
2	Alan To	6047776721	56 John Avenue East V5T 9E8	2022-09-02
3	Smith Jos Chan	7788006547	389 Ottawa Road V5W 2G9	2023-01-01
4	Nancy Stairs	2345698708	Flat 1204, 80 Silver Avenue V5E 3T2	2023-01-01
5	Hello World	1000101010	B019, 100 49th Avenue West	2023-03-01

Service

SID	Name	Cost
1	Basic checkup	30
2	Basic cleaning	50
3	Filling (one tooth)	100
4	Filling (two teeth)	200
5	Filling (three or more teeth)	300
6	Dental guard (per each)	200
7	Follow-up	0

Patient

PID	Name	Phone	Address	DateJoined
1	Adam Smith	9876543210	456 Money Street V5R 8T1	2022-10-01
2	Sherlock Holmes	8487950168	221B Baker Street	2022-10-01
3	John Stuart Mill	8791021068	13 Rodney Street	2023-02-05
4	John Keynes	7894805165	1883 Economics Road	2023-02-28

Appointment

AID	Date	Time	Duration	DID	PID
1	2022-10-01	09:30	00:45	1	1
2	2022-11-15	14:15	01:00	2	2
3	2022-11-15	14:15	00:30	1	1
4	2023-02-14	16:00	01:00	3	4
5	2023-03-01	10:30	00:45	4	3
6	2023-03-01	11:00	01:30	2	2
7	2023-03-01	15:00	01:00	3	4

Insurance_Policy

Insurer	PolicyNo
Pacific Blue Cross	000001
Pacific Blue Cross	000020A
SunLife	123-5068
Canadian Insurance	B1234146
Canadian Insurance	B1234147

Payment

PaymentID	Date	is Paid By Patient	is Paid By Insurance	Patient Amount	Policy Amount	AID	Insurer	PolicyNo
1	2022-10-01	1	1	20	60	1	SunLife	123-5068
2	2022-11-15	0	1	**NULL**	100	2	Pacific Blue Cross	000001
3	2022-11-20	1	0	30	**NULL**	2	**NULL**	**NULL**
4	2023-02-14	1	1	20	180	4	Pacific Blue Cross	000020A
5	2023-02-14	1	1	10	20	4	Canadian Insurance	B1234146
6	2023-03-01	1	0	50	**NULL**	5	**NULL**	**NULL**
7	2023-03-01	0	1	**NULL**	20	4	Canadian Insurance	B1234146
8	2023-03-01	1	1	50	50	7	Canadian Insurance	B1234146
9	2023-03-01	1	0	10	**NULL**	7	**NULL**	**NULL**

Service_Involved_In_Appointment

SID	AID
1	1
2	1
1	2
3	2
7	3
2	4
6	4
2	5
5	6
1	7
3	7

Referral

RefereePID	ReferrerPID
3	1
4	1