**Project Part 2**

Regina Powers, Ben Borszcz, Lohith Maralla, Bryan Vales

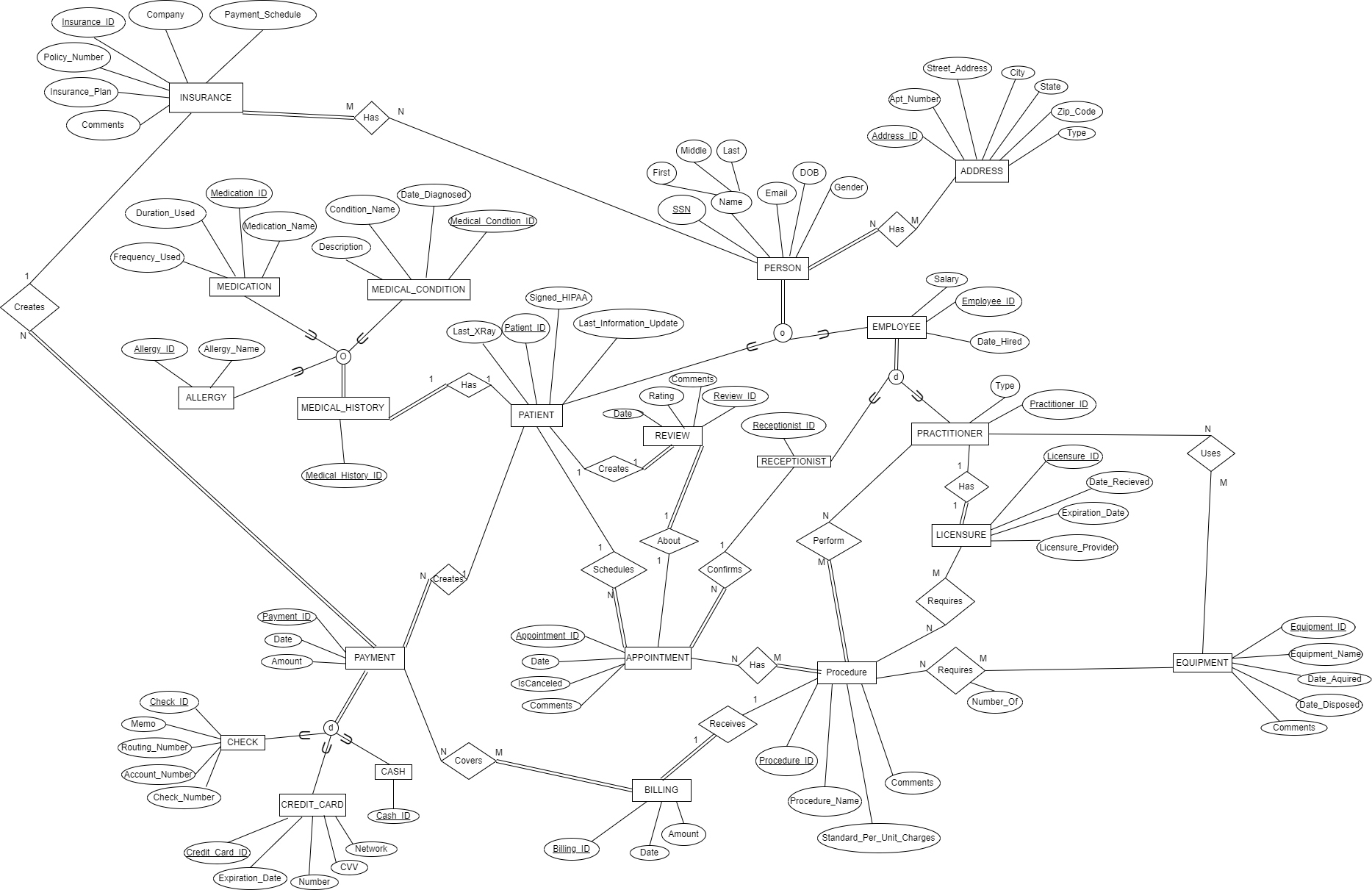
Dr. Hope Smilow is very pleased with the progress your team is making and with your initial proposal for her DB. Now it is the time to solidify the logical design of your database and to think how you are going to create queries to help her with information retrieval and making use of the data in her DB.

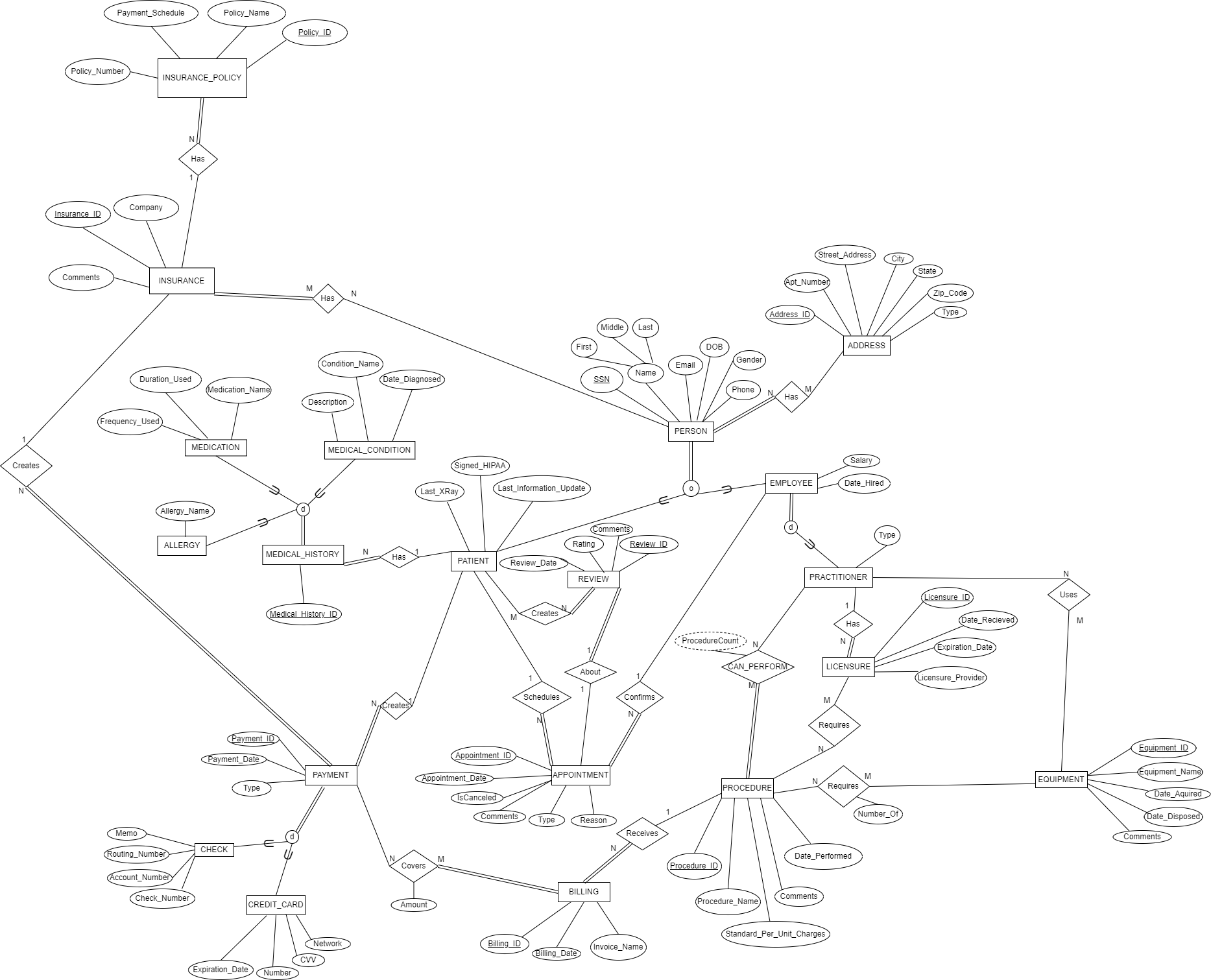
1. Review feedback provided for your initial design from PART 1 and make necessary changes. Your (E)ERD should be fully correct and ready to be mapped into the schema. Remember, if you start with an even partially incorrect diagram and create a model based on that, you will not have a working DB and all your work from this point forward may be faulty. It is extremely important to have a fully correct ERD before starting schema development.

**Feedback:**

1. Why do we have a receptionist as a separate specialization?
   1. DONE; removed receptionist
2. No need to connect Appt with Procedure. You should have appt type/complaint/reason but you do not know procedure codes to be used upfront.
   1. DONE (did not add complaint as this is in comment or review)
3. Medical History: the same record cannot be an Allergy and Medication at the same time
   1. DONE, changed o to d
4. No need for Cash specialization, there is no such item as Cash ID.
   1. DONE; removed
5. Specializations do not have PKs.
   1. DONE; removed all specialization PKs
6. Get clear understanding about what an Insurance Company Vs. Insurance Policy and how your DB should store this info.
   1. DONE; made insurance policy its own entity
7. Keep an eye on attributes that describe M:N relationships such as quantities, dates, amounts that depend on both sides. Specifically, Invoice:Procedure -- most important!
   1. procedure has billing changed to M:N
   2. patient creates review changed to M:N
   3. practitioner has licensure changed to 1:N
   4. patient has medical history changed to 1:N
8. The PaymentAmount attribute in Payment is not in the right place. Same with Billian Amount -- possibly derived or in the wrong place.
   1. DONE: moved to ‘covers’ relation
9. Medical EMps can perform procedures based on licensure type.
   1. Practitioner performs procedure → Practitioners can\_perform procedure
10. Creating Practitioner: Procedure pairs are not correct. You will have 100s of procedure codes. WIll you pair each one with individual employee records?
11. Additional changes were made by adding attributes were found to be missing

**Original ERD**



**Corrected/New ERD**

1. Using sentence notation learned in class, map your (E)ER diagram to a relational schema. Indicate all primary and foreign keys and show how they relate to each other. Make sure that you properly follow the mapping algorithm and evaluate and map each element shown in your ERD. Your relational schema must be fully consistent with your ERD. Show and explain all the steps you take in the process. Carefully review primary keys and relations/attribute names and make changes if necessary. Use your best judgment when mapping specializations and try to avoid NULLs.

Note that underlines indicate a primary key, while green italicized underlined text indicates a foreign key.

**Step 1: Map each regular entity type into a relation**

Each regular entity gets its own relation.

PROCEDURE(Procedure\_ID, Procedure\_Name, Standard\_Per\_Unit\_Charges, Comments, Dare\_Performed)

EQUIPMENT(Equipment\_ID, Equipment\_Name, Date\_Aquired, Date\_Disposed, Comments)

ADDRESS(Address\_ID, Apt\_Number, Street\_Address, City, State, Zip\_Code, Type)

INSURANCE(Insurance\_ID, Company, Comments)

**Step 2: Map weak entities**

We have no weak entities.

**Step 3: Mapping of 1:N relation**

Theprimary key of the 1 side is added as a foreign key to the N side.

PAYMENT(Payment\_ID, Payment\_date, *Insurance\_ID, SSN*)

BILLING(Billing\_ID, Billing\_date, *Procedure\_ID*)

LICENSURE(Licensure\_ID, Date\_Recieved, Expiration\_Date, Licensure\_Provider, *SSN*)

APPOINTMENT(Appointment\_ID, Appointment\_date, IsCanceled, Comments, Type, Reason, *EmployeeSSN, PatientSSN*)

MEDICAL\_HISTORY(Medical\_History\_ID, *SSN*)

INSURANCE\_POLICY(Policy\_ID, Payment\_Schedule, Policy\_Number, Policy\_Name, *Insurance\_ID*)

**Step 4: Mapping of 1:1 relation**

The primary key from the partial participation is added to the full participation as a foreign key

REVIEW(Review\_ID, Review\_date, Rating, Comments, Appointment\_ID)

**Step 5: Mapping of M:N relation**

A new relation is made for each M:N relation

PERSON\_HAS\_INSURANCE(SSN, Insurance\_ID)

PERSON\_HAS\_ADDRESS(SSN, Address\_ID)

PAYMENT\_COVERS\_BILLING(Payment\_ID, Billing\_ID, Amount)

PROCEDURE\_REQUIRES\_EQUIPMENT(Procedure\_ID, Equipment\_ID, Number\_Of)

PROCEDURE\_REQUIRES\_LICENSURE(Procedure\_ID, Licensure\_ID)

PRACTITIONER\_CAN\_PERFORM\_PROCEDURE(SSN, Procedure\_ID, ProcedureCount)

PRACTITIONER\_USES\_EQUIPMENT(SSN, Equipment\_ID)

PATIENT\_CREATES\_REVIEW(SSN, Review\_ID)

**Step 6: Mapping of Multivalued Attributes**

We have no multivalued attributes.

**Step 7: Mapping specializations/generalizations**

We chose the option to create multiple relations for all super/sub classes, where we included the foreign key of superclasses in the subclasses. Indentation represents relations being a specialization.

MEDICAL\_HISTORY(Medical\_History\_ID, *SSN*)

MEDICATION(*Medical\_History\_ID*, Medication\_Name, Frequency\_Used, Duration\_Used)

MEDICAL\_CONDITION(*Medical\_History\_ID*, Condition\_Name, Description, Date\_Diagnosed)

ALLERGY(*Medical\_History\_ID*, Allergy\_Name)

PERSON(SSN, First, Middle, Last, Email, DOB, Gender, Phone)

EMPLOYEE(*SSN*, Salary, Date\_Hired)

PRACTITIONER(*SSN*, Type)

PATIENT(*SSN*, Last\_XRay, Signed\_HIPAA, Last\_Information\_Update)

PAYMENT(Payment\_ID, Payment\_date, Type *Insurance\_ID, SSN*)

CREDIT\_CARD(*Payment\_ID*, Expiration\_Date, Number, CVV, Network)

CHECK(*Payment\_ID*, Memo, Routing\_Number, Account\_Number, Check\_Number)

1. Given your relational schema, provide the relational algebra to perform the following queries. If your schema cannot provide answers to these queries, revise your ER Model and your relational schema to contain/supply the appropriate information for these queries. Double check that all required information can be calculated by using RA and you do not store or utilize derived attributes.

Note that we are using **natural join** because we want to match attributes that have the same name. In our case, the foreign keys have the same name as the primary keys they are related to, allowing us to use natural join.

* 1. Create a list of patients and the medications they currently take

Π First, Last, Medication\_Name((PATIENT \* PERSON) \* MEDICATION)

* 1. Display patient information for patients who currently have Delta Dental insurance policy.

Π First, Middle, Last, SSN, Email, DOB, Gender, Patient\_ID, Signed\_HIPAA, Last\_Xray, Last\_Information\_Update (σCompany= ‘Delta Dental’(((PATIENT \* PERSON) \* PERSON\_HAS\_INSURANCE) \*INSURANCE)

* 1. Generate a list of procedures and dates of service performed by doctor Smilow. ΠProcedure\_Name, Datetime\_Performed (σlast=Smilow((PRACTITIONER\*PRACTITIONER\_CAN\_PERFORM\_PROCEDURE) \*PROCEDURE))
  2. Print out a list of past due invoices with patient contact information. Past due is defined as over 30 days old with a balance over $10.

ΠInvoice\_Name,Amount, Date, First, Middle, Last, Phone, Email (σ(Amount > 10) AND (GETDATE() - Billing\_date > 30)((((BILLING\* PAYMENT\_COVERS\_BILLING) \* PAYMENT) \*PATIENT) \*PERSON)

* 1. Find the patients who brought the most revenue in the past year.

TOTAL\_AMOUNTS ← SSN FSUM Amount(((PAYMENT\_COVERS\_BILLING\*PAYMENT)\*PATIENT)\*PERSON)

MAX\_AMOUNTS ← SSN FMAX Sum\_Amount(TOTAL\_AMOUNTS)

ΠFirst,Middle,Last, Max\_Amount(MAX\_AMOUNTS)

* 1. Create a list of doctors who performed less than 5 procedures this year.

ΠFirst, Middle, Last(σProcedureCount<5(((PROCEDURE \*PRACTITIONER\_CAN\_PERFORM\_PROCEDURE) \* PRACTITIONER) \* EMPLOYEE) \* PERSON))

* 1. Find the highest paying procedures, procedure price, and the total number of those procedures performed.

HIGHEST\_PROCEDURES ← Procedure\_Name FMAX Standard\_Per\_Unit\_Charges (PROCEDURE)

NUM\_PROCEDURES ← Procedure\_Name FCOUNT Procedure\_Name (PROCEDURE)

ΠProcedure\_Name, Standard\_Per\_Unit\_Charge, Count\_Procedure(HIGHEST\_PROCEDURES \* NUM\_PROCEDURES)

* 1. Create a list of all payment types accepted, number of times each of them was used, and total amount charged to that type of payment.

AMOUNT\_CHARGED ← Type FSUM Amount (PAYMENT\*PAYMENT\_COVERS\_BILLING)

TIMES\_USED ← Type FCOUNT Amount(PAYMENT\*PAYMENT\_COVERS\_BILLING)

ΠType, Sum\_Amount, Count\_Amount (AMOUNT\_CHARGED \* TIMES\_USED )

* 1. List ids and names of insurance plans ever used by patients and how many patients have that plan.

Policy\_ID, Policy\_Name FCOUNT SSN(((INSURANCE\_POLICY\*INSURANCE)\*PERSON\_HAS\_INSURANCE)\*PERSON)

1. Provide three additional interesting queries in plain English and relational algebra. Each of your queries should include at least one of these. Queries should be interesting and involve multiple entities and operations.

1. Retrieving a person’s name and address who left a review below 5 stars:

SUBPAR\_REVIEW ← σRating<5 (REVIEW)

PERSON\_DETAILS ← ΠFirst, Middle, Last, ,Appointment\_Date, Street\_Address (((PATIENT \* PERSON)\*APPOINTMENT)\*ADDRESS)

PERSON\_BAD\_REVIEW ← PERSON\_DETAILS ⟕Appointment\_Date=Review\_Date SUBPAR\_REVIEW

2. Finds the highest paying position:

EMPLOYEE\_DETAILS ← ΠFirst, Middle, Last, Employee\_ID, Salary (EMPLOYEE \* PERSON)

PRACTIONER\_NAMES ← EMPLOYEE\_DETAILS⟗SSN= SSNPRACTITIONERS

MAX\_PROFESSION ← Type FMAX Salary(PRACTIONER\_NAMES)

3. Lists the employees and the amount of equipment they have used:

PRACTIONER\_DETAILS ← ΠFirst, Middle, Last, ((EMPLOYEE \*PRACTITIONER)\* PERSON)

Equipment\_Name FCOUNT SSN(PRACTIONER\_DETAILS\*EQUIPMENT)

1. Now we are almost ready to implement our DB in RDBMS (SQL). Before we can do this, you need to complete specification sheets for each relation (metadata). This information would be passed to the programmers and used to create SQL code. List all your proposed relations. They are about to become DB tables. For each table, describe its purpose. List all attributes. For each attribute specify constraints (NULL, semantic, length, proposed data type, etc.) For FKs, identify matching relations and PKs and specify any relevant referential integrity rules. Do NOT include any parts of actual SQL code here. This documentation should be language independent.

Note that proposed data types are listed immediately after the attribute name, prior to the attributes description.

1. PROCEDURE:
   1. Purpose: stores information about medical procedures that can be performed by practitioners.
   2. Attributes:
      1. Procedure\_ID (PK): INT; unique identifier for the procedure.
      2. Procedure\_Name: VARCHAR; name of the procedure.
      3. Standard\_Per\_Unit\_Charges: INT; standard charge for each unit of the procedure.
      4. Comments: VARCHAR; additional comments about the procedure.
      5. Date\_Performed: DATE; date the procedure was performed.
2. EQUIPMENT:
   1. Purpose: stores information about medical equipment that can be used by practitioners.
   2. Attributes:
      1. Equipment\_ID (PK): INT; unique identifier for the equipment.
      2. Equipment\_Name: VARCHAR; name of the equipment.
      3. Date\_Acquired: DATE; date the equipment was acquired.
      4. Date\_Disposed: DATE; date the equipment was disposed of.
      5. Comments: VARCHAR; additional comments about the equipment.
3. ADDRESS:
   1. Purpose: stores information about addresses associated with patients and practitioners.
   2. Attributes:
      1. Address\_ID (PK): INT; unique identifier for the address.
      2. Apt\_Number: INT; apartment number associated with the address.
      3. Street\_Address: VARCHAR; street address associated with the address.
      4. City: VARCHAR; city associated with the address.
      5. State: VARCHAR; state associated with the address.
      6. Zip\_Code: INT; ZIP code associated with the address.
      7. Type: VARCHAR; type of address (e.g., "home", "work").
4. INSURANCE:
   1. Purpose: stores information about insurance companies.
   2. Attributes:
      1. Insurance\_ID (PK): INT; unique identifier for the insurance company.
      2. Company: VARCHAR; name of the insurance company.
      3. Comments: VARCHAR; additional comments about the insurance company.
5. PAYMENT:
   1. Purpose: stores information about payments made by patients.
   2. Attributes:
      1. Payment\_ID (PK): INT; unique identifier for the payment.
      2. Payment\_date: DATE; date the payment was made.
      3. Insurance\_ID (FK): INT; foreign key that references the insurance company associated with the payment.
      4. SSN (FK): INT; foreign key that references the patient who made the payment.
6. BILLING:
   1. Purpose: stores information about procedures performed on patients and associated charges.
   2. Attributes:
      1. Billing\_ID (PK): INT; unique identifier for the billing.
      2. Billing\_date: DATE; date the billing was issued.
      3. Procedure\_ID (FK): INT; foreign key that references the procedure associated with the billing.
7. LICENSURE:
   1. Purpose: stores information about professional licensures held by practitioners.
   2. Attributes:
      1. Licensure\_ID (PK): INT; unique identifier for the licensure.
      2. Date\_Received: DATE; date the licensure was received.
      3. Expiration\_Date: DATE; date the licensure expires.
      4. Licensure\_Provider: VARCHAR; provider of the licensure.
      5. SSN (FK): INT; foreign key that references the practitioner associated with the licensure.
8. APPOINTMENT:
   1. Purpose: stores information about appointments made by patients with practitioners.
   2. Attributes:
      1. Appointment\_ID (PK): INT; unique identifier for the appointment.
      2. Appointment\_date: DATE; date of the appointment.
      3. IsCanceled: BOOLEAN; flag indicating whether the appointment was canceled.
      4. Comments: VARCHAR; additional comments about the appointment.
      5. Type: VARCHAR; type of appointment (e.g., "checkup", "consultation").
      6. Reason: VARCHAR; reason for the appointment.
      7. EmployeeSSN (FK): INT;foreign key that references the practitioner who will see the patient.
      8. PatientSSN (FK): INT; foreign key that references the patient who made the appointment.
9. MEDICAL\_HISTORY
   1. Purpose: stores information about the proposed relations for the relational schema:
   2. Attributes
      1. Medical\_History\_ID (PK): INT; unique identifier for the medical history.
      2. SSN (FK): INT; foreign key that references the patient associated with the medical history
10. INSURANCE\_POLICY:
    1. Purpose: stores information about insurance policies held by patients.
    2. Attributes:
       1. Policy\_ID (PK): INT; unique identifier for the insurance policy.
       2. Payment\_Schedule: VARCHAR; payment schedule for the policy.
       3. Policy\_Number: INT; policy number.
       4. Policy\_Name: VARCHAR; name of the policy.
       5. Insurance\_ID (FK): INT;foreign key that references the insurance company associated with the policy.
11. REVIEW:
    1. Purpose: stores information about reviews made by patients about practitioners.
    2. Attributes:
       1. Review\_ID (PK): INT; unique identifier for the review.
       2. Review\_date: DATE; date of the review.
       3. Rating: INT; rating (e.g., 1-5 stars) given in the review.
       4. Comments:VARCHAR; additional comments about the review.
       5. Appointment\_ID (FK): INT; foreign key that references the appointment associated with the review.
12. PERSON:
    1. Purpose: stores basic information about individuals (both patients and practitioners).
    2. Attributes:
       1. SSN (PK):INT; unique identifier for the individual.
       2. First: VARCHAR; first name of the individual.
       3. Middle: VARCHAR; middle name of the individual.
       4. Last: VARCHAR; last name of the individual.
       5. Email: VARCHAR: email address of the individual.
       6. DOB: DATE; date of birth of the individual.
       7. Gender: VARCHAR; gender of the individual.
       8. Phone: INT; phone number of the individual.
13. EMPLOYEE:
    1. Purpose: stores information about practitioners who are also employees of the medical facility.
    2. Attributes:
       1. SSN (PK, FK): INT; foreign key that references the practitioner associated with the employee.
       2. Salary: INT; salary of the employee.
       3. Date\_Hired: DATE;date the employee was hired.
14. PRACTITIONER:
    1. Purpose: stores information about practitioners.
    2. Attributes:
       1. SSN (PK, FK): INT; unique identifier for the practitioner.
       2. Type: VARCHAR; type of practitioner (e.g., "physician", "nurse").
15. PATIENT:
    1. Purpose: stores information about patients.
    2. Attributes:
       1. SSN (PK, FK): INT; unique identifier for the patient.
       2. Last\_XRay: DATE; date of the patient's last X-ray.
       3. Signed\_HIPAA: BOOLEAN; flag indicating whether the patient has signed a HIPAA agreement.
       4. Last\_Information\_Update: date of the patient's last information update.
16. CREDIT\_CARD:
    1. Purpose: stores information about credit card payments made by patients.
    2. Attributes:
       1. Payment\_ID (PK, FK): INT; foreign key that references the payment associated with the credit card payment.
       2. Expiration\_Date: DATE; expiration date of the credit card.
       3. Number: INT; credit card number.
       4. CVV: INT; CVV code of the credit card.
       5. Network: VARCHAR; network (e.g., "Visa", "Mastercard") of the credit card.
17. CHECK:
    1. Purpose: stores information about check payments made by patients.
    2. Attributes:
       1. Payment\_ID (PK, FK): INT; foreign key that references the payment associated with the check payment.
       2. Memo: VARCHAR; memo associated with the check payment.
       3. Routing\_Number: INT; routing number of the check.
       4. Account\_Number: INT; account number of the check.
       5. Check\_Number: INT; check number.
18. PERSON\_HAS\_INSURANCE:
    1. Purpose: stores the association between a person (patient or practitioner) and an insurance company.
    2. Attributes:
       1. SSN (PK, FK): INT; foreign key that references the person associated with the insurance policy.
       2. Insurance\_ID (PK, FK): INT; foreign key that references the insurance company associated with the insurance policy.
19. PERSON\_HAS\_ADDRESS:
    1. Purpose: stores the association between a person (patient or practitioner) and an address.
    2. Attributes:
       1. SSN (PK, FK): INT; foreign key that references the person associated with the address.
       2. Address\_ID (PK, FK): INT; foreign key that references the address associated with the person.
20. PAYMENT\_COVERS\_BILLING:
    1. Purpose: stores the association between a payment and a billing.
    2. Attributes:
       1. Payment\_ID (PK, FK): INT; foreign key that references the payment associated with the billing.
       2. Billing\_ID (PK, FK): INT; foreign key that references the billing associated with the payment.
       3. Amount: amount paid by the payment for the billing.
21. PROCEDURE\_REQUIRES\_EQUIPMENT:
    1. Purpose: stores the association between a procedure and the equipment required to perform it.
    2. Attributes:
       1. Procedure\_ID (PK, FK): INT; foreign key that references the procedure associated with the equipment.
       2. Equipment\_ID (PK, FK): INT; foreign key that references the equipment required for the procedure.
       3. Number\_Of: INT; number of units of the equipment required for the procedure.
22. PROCEDURE\_REQUIRES\_LICENSURE:
    1. Purpose: stores the association between a procedure and the licensure required to perform it.
    2. Attributes:
       1. Procedure\_ID (PK, FK): INT; foreign key that references the procedure associated with the licensure.
       2. Licensure\_ID (PK, FK): INT; foreign key that references the licensure required for the procedure.
23. PRACTITIONER\_CAN\_PERFORM\_PROCEDURE:
    1. Purpose: stores the association between a practitioner and the procedures they can perform.
    2. Attributes:
       1. SSN (PK, FK): INT; foreign key that references the practitioner associated with the procedure.
       2. Procedure\_ID (PK, FK): INT; foreign key that references the procedure that the practitioner can perform.
       3. ProcedureCount: INT; number of times the practitioner has performed the procedure.
24. PRACTITIONER\_USES\_EQUIPMENT:
    1. Purpose: stores the association between a practitioner and the equipment they use.
    2. Attributes:
       1. SSN (PK, FK): INT; foreign key that references the practitioner associated with the equipment.
       2. Equipment\_ID (PK, FK): INT; foreign key that references the equipment used by the practitioner.
25. PATIENT\_CREATES\_REVIEW:
    1. Purpose: stores the association between a patient and the reviews they create.
    2. Attributes:
       1. SSN (PK, FK): INT; foreign key that references the patient associated with the review.
       2. Review\_ID (PK, FK): INT; foreign key that references the review created by the patient.

The constraints for the attributes in each table would be as follows:

* All primary key attributes cannot be NULL.
* Foreign key attributes must match the data type and length of the primary key attributes they reference.
* For other attributes, constraints would depend on the specific requirements of the application. For example, date attributes could have constraints to ensure they are in a valid format, and text attributes could have a maximum length to prevent overflows. Numeric attributes could have constraints to ensure they are within a valid range.

Referential integrity rules would be applied to maintain consistency between tables. For example, in the PAYMENT table, the foreign keys SSN and Insurance\_ID would reference the SSN and Insurance\_ID primary keys in the PERSON and INSURANCE tables, respectively. If a payment is associated with a person or insurance company that does not exist in the corresponding table, the referential integrity rule would prevent the insertion or update of that payment record.

Another example of referential integrity rules would be in the APPOINTMENT table, where EmployeeSSN and PatientSSN would reference the SSN primary key in the PRACTITIONER and PATIENT tables, respectively. If an appointment is associated with a practitioner or patient that does not exist in the corresponding table, the referential integrity rule would prevent the insertion or update of that appointment record.

1. Submit a professionally written and well formatted report showing ALL your work. Include your original ERD and feedback from PART 1. List all team member contributions. Provide any relevant details on how your team has been functioning.

Regina Powers: Completion of feedback from part 1, relational schema, relational algebra, proposed data types of meta-data.

Lohith Maralla: Relational Schema, creation of queries, relational algebra

Bryan Vales: Revision of ERD, Mapping into Relational Schema, Relational Algebra

Ben Borszcz: Relational Schema, Meta-Data, Relational Algebra

Our team continues to be functioning well. We meet a few times to work together and figure out how we want to work on the project individually. We all check each other’s work to make sure that we are all content with what is being turned in. We utilize the professor for any questions we have, and get feedback early on to make sure that we are on the correct track.

1. Save all your work as you will need to use it for the next phase of the project.