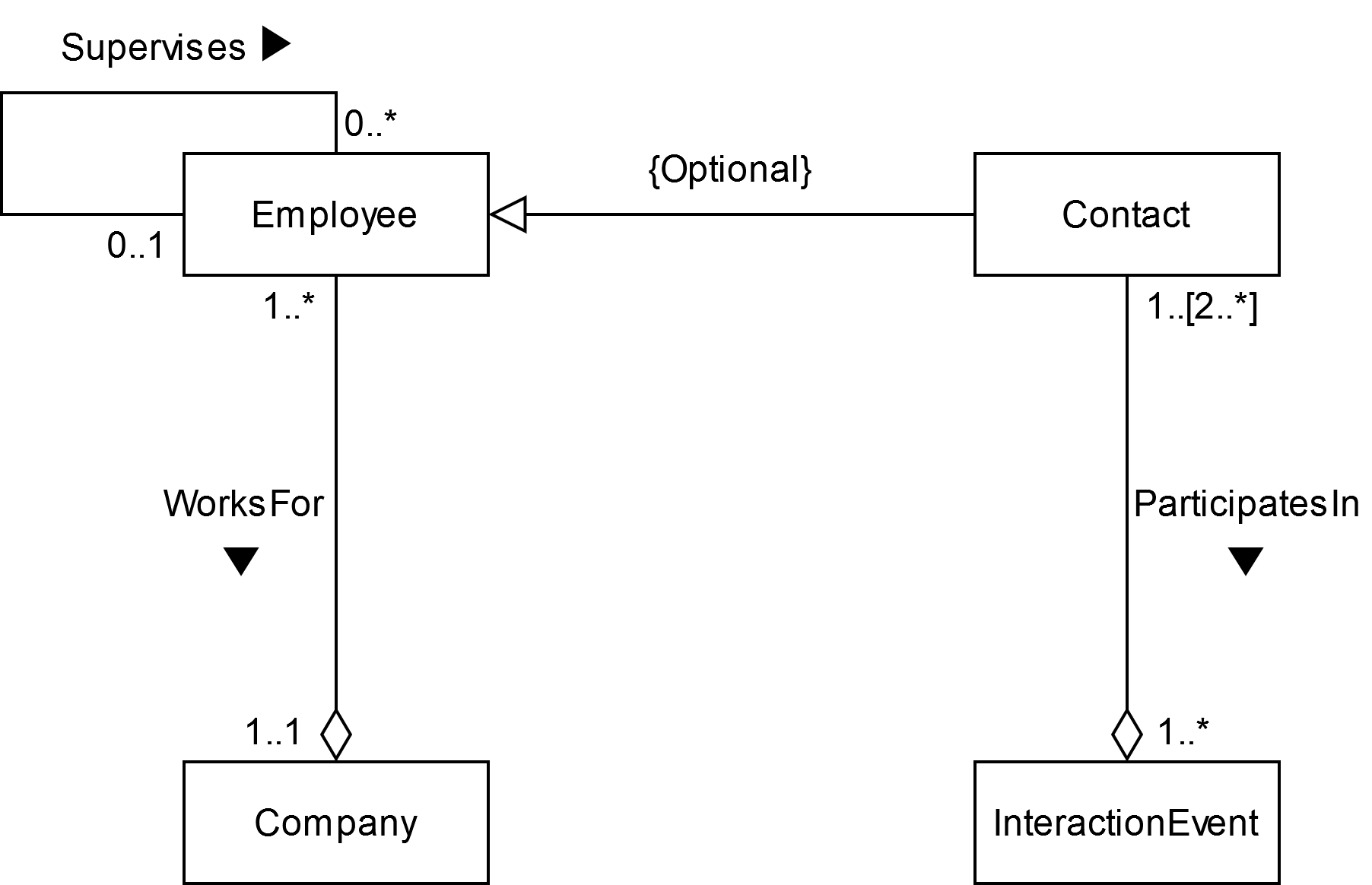
**Business Contact Interaction Tracking System Project Report**

**Project Overview**

Throughout the second half of this semester, I designed, developed, and implemented in PostgreSQL a relational database that records information concerning communications between business contacts. The database holds data, including the type, date, and comments, about each interaction between contacts. In addition, the database also stores data about each contact. This data consists of the contact’s name, salutation, job title, employer, manager, subordinates, secretary, address(es), phone number(s), email address(es), and personal comments. Moreover, the database contains company information, such as company names. A unique alphanumeric string identifies each interaction event, contact, or company in the system. Given that the records are correctly inserted into the database, the user can perform SQL queries to search the data based on specific criteria.

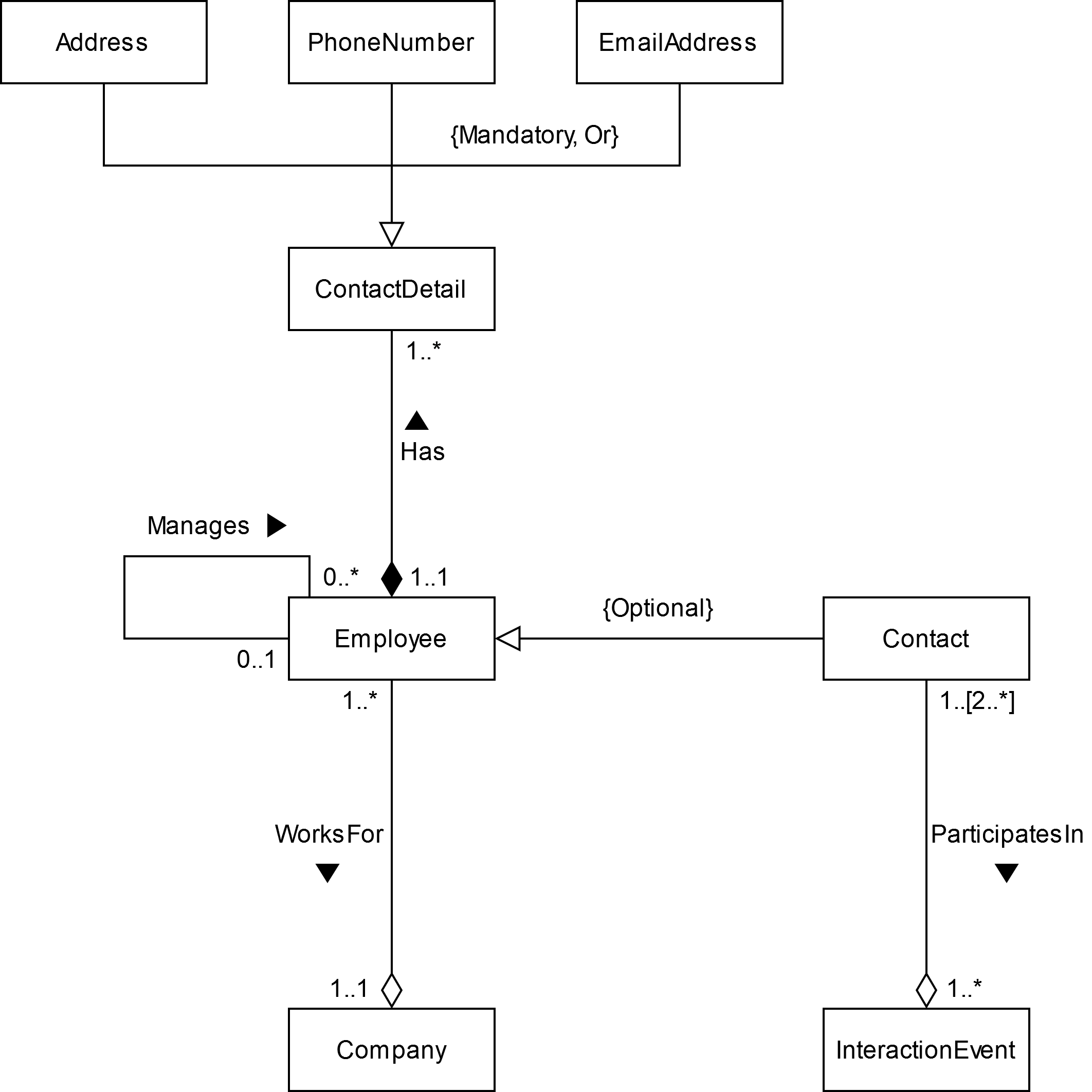
**Conceptual Design**

As the first stage of the project, I developed a conceptual data model. This stage consisted of identifying the required entities, relationships, attributes, and constraints and recording them in a data dictionary. To determine this information, I studied the case study description. I initially identified three entity types: Contact, InteractionEvent, and Company. However, as the database also records data for other employees, such as a contact’s subordinates and secretary, of a company, I decided to create another entity type: Employee. Since business contacts are employees of a company, I determined that Contact is a subclass of Employee. Given these four entity types, I then identified the relationship types between them as well as each entity type’s attributes based on the case study document. For every attribute, I specified its data type, data size, and NULL constraint in the data dictionary. As the last step of the conceptual design stage, I constructed an enhanced entity-relationship (EER) diagram representing the entities, relationships, and multiplicity constraints.



**Logical Design**

The next stage in the database development process was developing a logical data model. Based on feedback I received regarding my original conceptual design, I revised my conceptual data model to include four additional entities: ContactDetail, Address, PhoneNumber, and EmailAddress. ContactDetail is a superclass from which the subclasses Address, PhoneNumber, and EmailAddress inherit. Every entity occurrence of ContactDetail must also be an occurrence of Address, PhoneNumber, or EmailAddress. As a result, the inheritance relationship has mandatory participation and disjoint constraints. By storing contact information in separate entities instead of solely the Contact entity, I eliminated the repeating groups of attributes present in my original design. As part of my refined conceptual design, I also modified the EER diagram to include the added entity types.



Using the refined conceptual data model, I derived relations from the entity and relationship types according to the guidelines in the textbook. In general, the relations and corresponding attributes are similar to those in the data dictionary but with the addition of foreign key attributes representing the relationships between the relations. Nevertheless, due to the nature of the Manages and ParticipatesIn relationships, I created additional relations to represent these relationships. For instance, since a manager and their subordinate must work at the same company, companyID was transitively dependent on userID via managerID in the Employee relation. To eliminate this transitive dependency, I created the Subordinate relation that relates a subordinate to their manager and enforces the constraint that both users must have the same companyID. Furthermore, because multiple contacts can participate in a single interaction event and a single contact can participate in multiple interaction events, I created another relation, InteractionEventParticipation, that links a contact to the interaction events that the contact participated in. I documented each relation, together with its attributes, primary key, and foreign key(s), in a table.

Before proceeding to the implementation stage, I validated the logical data model. I first validated the model using normalization. Specifically, I examined every relation to check that it was in Boyce-Codd Normal Form (BCNF). I then validated the model against sample user transactions using the example searches listed on the first page of the case study document. By validating the logical model, I ensured that I constructed a model that supports the required transactions, reduces data redundancy, and addresses insert, update, and deletion anomalies.

**PostgreSQL Implementation**

Table Creation

After completing the logical design stage, I began implementing the database in PostgreSQL by creating a table with the associated integrity constraints for each relation documented in the logical data model.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

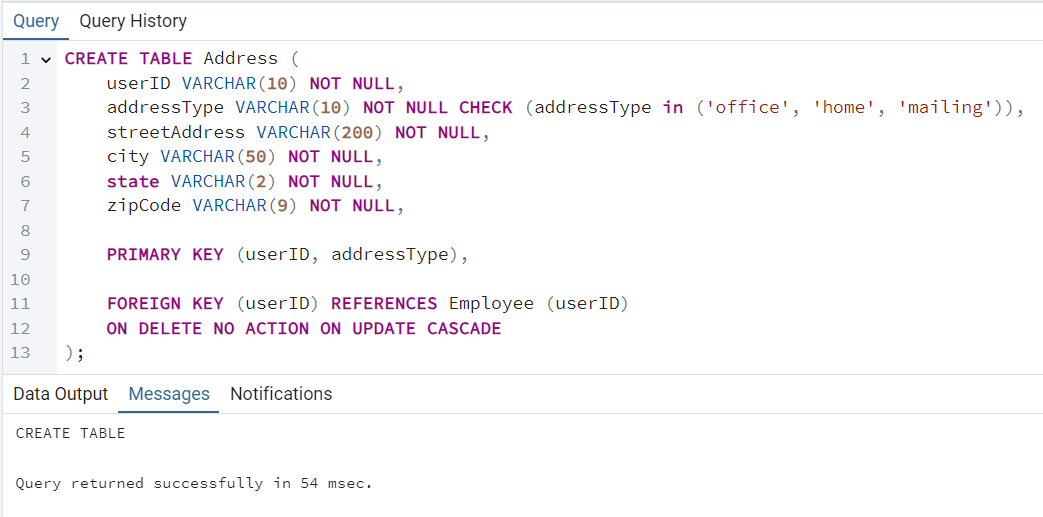
Description automatically generated

A screenshot of a computer

Description automatically generated

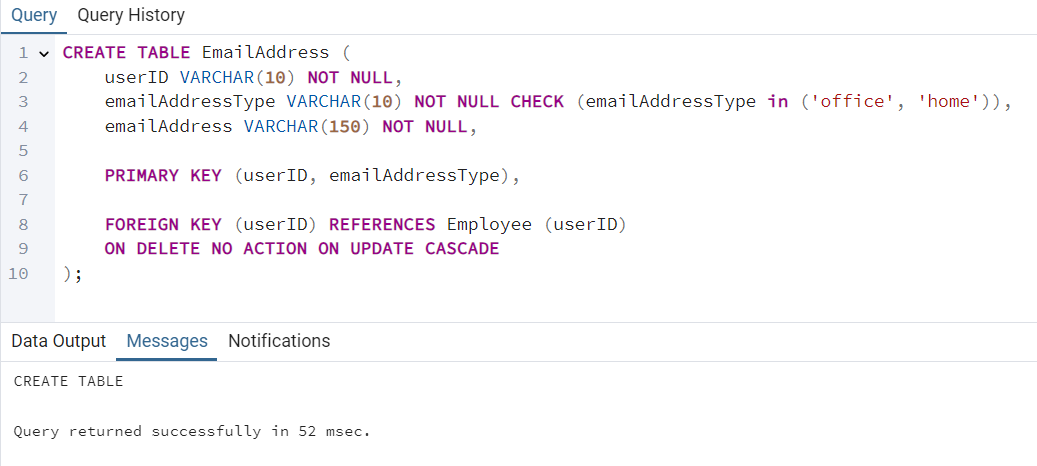
A screenshot of a computer program

Description automatically generated



A screenshot of a computer code

Description automatically generated



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Data Insertion

Once I created all the database tables, I then inserted numerous rows of fictional data into each table.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer code

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer screen

Description automatically generated

Queries

Finally, I wrote and executed the SQL code for the ten sample queries listed in the case study.

Query 1: List all the interaction events (date, event type, comments) associated with a particular contact

A screenshot of a computer

Description automatically generated

Query 2: List all the interaction events made during a particular time period, using MM/DD/YYYY as the input format for date

A screenshot of a computer

Description automatically generated

Query 3: List the information for all contacts available in a particular company

A screenshot of a computer program

Description automatically generated

Query 4: List the total number of phone calls made during a particular time period, using MM/DD/YYYY as the input format for date

A screenshot of a computer

Description automatically generated

Query 5: List all the subordinates (name, job title, office phone number, primary email address) of a particular contact

A screenshot of a computer program

Description automatically generated

Query 6: List all contacts that have more than 2 subordinates and a secretary

My database does not contain any contacts with more than 2 subordinates and a secretary, so this query did not return any rows.

A screenshot of a computer

Description automatically generated

Query 7: List all companies and the number of contacts associated with them, sorted in decreasing order by the number of contacts

A screenshot of a computer

Description automatically generated

Query 8: List all contacts with greater than 3 interaction events

A screenshot of a computer

Description automatically generated

Query 9: List the total number of phone call events, total number of email events, and total number of postal mail events

A screenshot of a computer

Description automatically generated

Query 10: List all contacts with a home address in a particular city

A screenshot of a computer

Description automatically generated

**Conclusion**

Over the course of this project, I designed, developed, and implemented a relational database that tracks interactions between business contacts. To complete this project, I used a three-stage process in which each stage I built on my work from the previous step. By refining, normalizing, and validating the design before translating the model into a PostgreSQL database, I reduced data redundancy, ensured data integrity, and addressed potential issues in the database implementation. From this project, I learned how to construct a data model based on a case study, how to create a database in PostgreSQL, and how to query the database to search for desired information. Additionally, I also practiced using SQL as both a data definition language and a data manipulation language. Overall, by working on this project, I gained valuable experience in database design and learned the importance of planning in the development process.