AUTOMATA Components Tracker: Streamlining Vehicle Production Through Enhanced Inventory and Order Management

Brandon Barber: Lead Developer

Michael Ridgeway: Project Manager

Li Zhu: Analyst

College of IST, Pennsylvania State University, University Park, PA, USA.

DS 220 – Team Projects on Relational Database Design & Implementation

Compiled by Dr. Mathias Fonkam (Instructor)

Automata, Inc. specializes in the production of specialty vehicles, a process that necessitates meticulous coordination across several departments, each responsible for a different vehicle type, such as limousines, trucks, vans, or RVs. A critical component of ensuring the smooth operation and timely production of these vehicles is the efficient management of vehicle parts – from inventory control to the procurement process. Automata's purchasing department plays a pivotal role in this ecosystem, managing orders for specific components needed by the production departments and maintaining a streamlined process for delivering these materials, either from inventory or through orders placed with suppliers. Given the complexity of managing numerous parts, multiple suppliers, and the varying needs of each production department, Automata, Inc. has recognized the necessity of developing a sophisticated database system. This system aims to enhance the visibility and management of orders, inventory levels, and supplier relationships, thereby accelerating the production process and reducing potential bottlenecks that could impact delivery timelines.

As the analyst in the team, my primary role involved extensive collaboration with teammates to spearhead the development of Automata, Inc.'s database system. This collaboration was pivotal in ensuring that all aspects of the database, from its conceptualization to its implementation, were tightly aligned with the project's objectives and Automata, Inc.'s operational needs.

I took the initiative to develop an Enhanced Entity-Relationship (EER) model, which serves as the backbone of our database system. This task commenced with an in-depth analysis of a conceptual EER diagram provided by a teammate. My responsibility was to refine this diagram further, ensuring it encapsulated all necessary entities, their attributes, and the intricate relationships among them. The EER diagram was meticulously crafted to include all relationships, clearly delineate Primary Keys (PKs) and Foreign Keys (FKs), and highlight the cardinality and participation constraints that dictate the dynamics of our database system.

Armed with a comprehensive understanding of the database structure and the operational intricacies of Automata, Inc., I formulated six critical queries. These queries were designed to address specific informational needs and operational scenarios faced by Automata, Inc. They span across inventory checks, order tracking, supplier management, and operational efficiency. Developing these queries required a deep dive into SQL, leveraging complex JOIN operations, aggregate functions, and conditional logic to extract meaningful insights from the database.

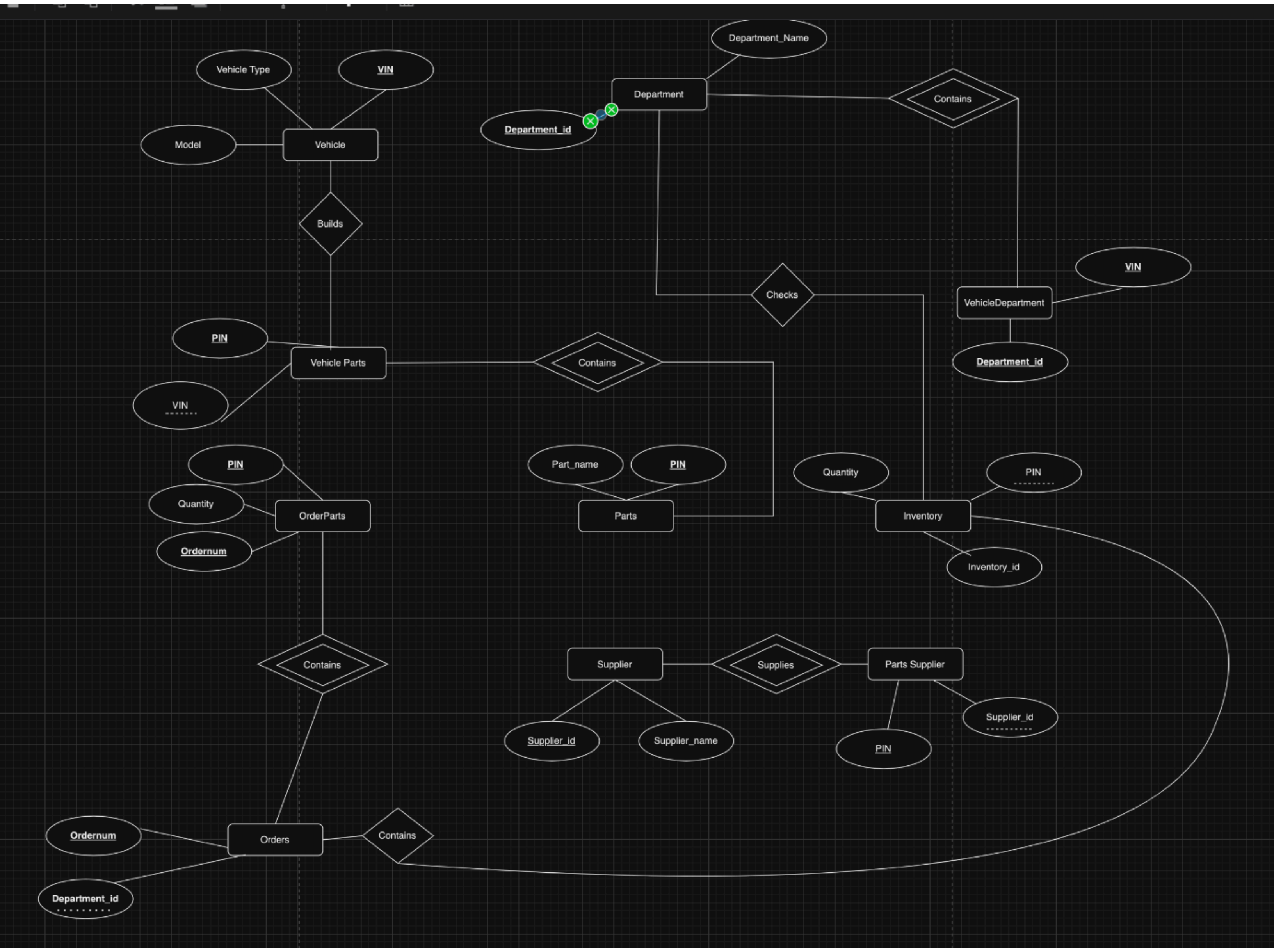
As the lead developer, my role was to lead the effort in implementing the previously created conceptual model, specifically the Enhanced Entity-Relationship model, into SQL. This step of database creation allows the database to be used as the purchasing department of Automata Inc. intended.

The steps taken to develop the SQL implementation of the database model involved using my knowledge of table creation in SQL to create tables that accurately reflected the relationships between entities. Upon creating the SQL implementation of the Automata Inc. database system my team was able to create and execute test queries tailored to the perceived uses of the database system.

As the project manager, my role was the organization of the group's communications and progress. I also assisted group members by implementing the initial conceptual model and then the logical model. I managed expectations from the TA and articulated those requirements to group members in order to finalize the deliverable.

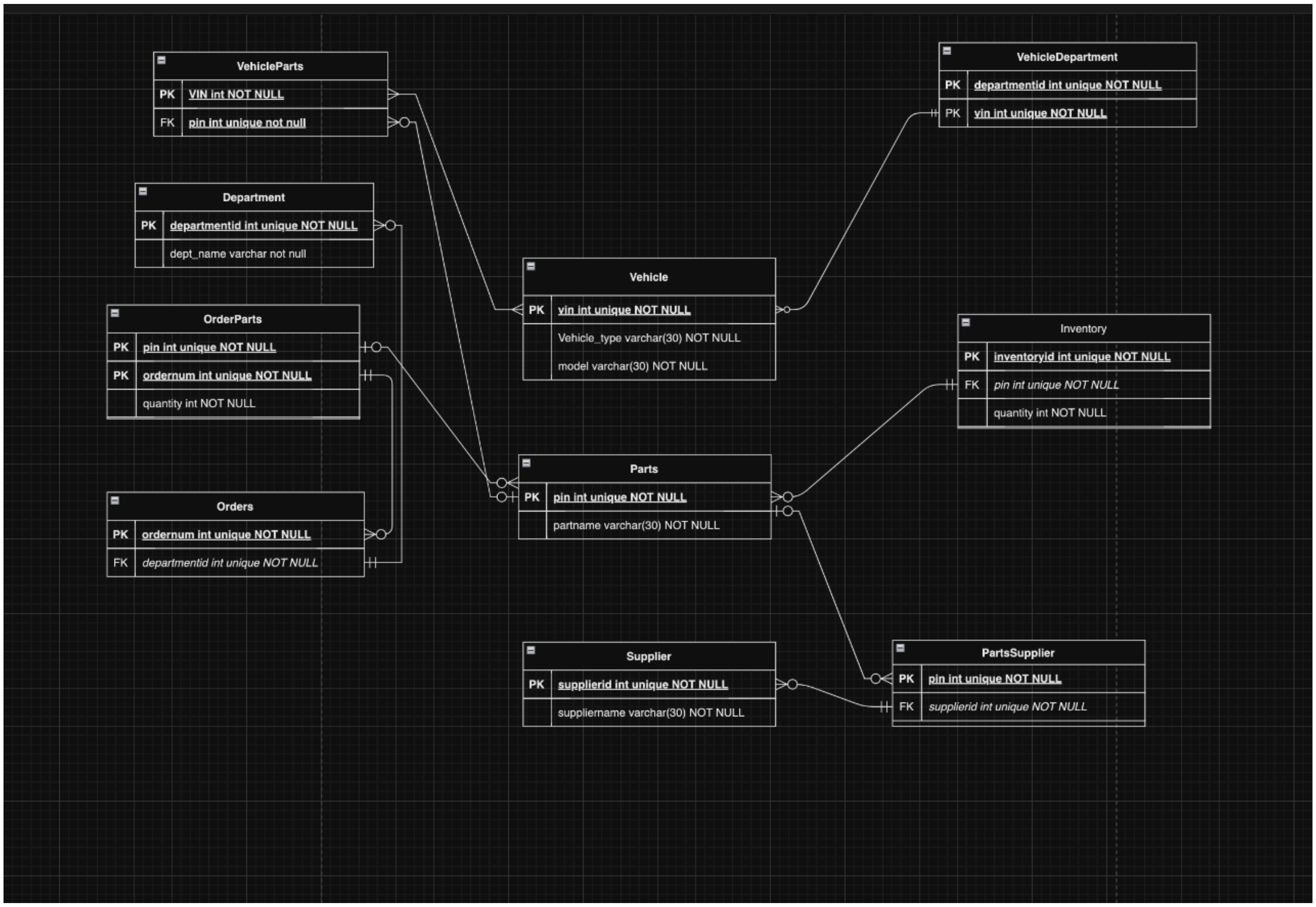
EER Model:

Automata, Inc., EER model serves as a blueprint for designing a database system capable of managing the intricacies of specialty vehicle production, including departmental orders, inventory tracking, and supplier management. This model outlines the core entities involved in Automata's operations, such as Departments, Orders, Items (Parts), Inventory, and Suppliers, along with the crucial relationships among them.



Logical Data Model:

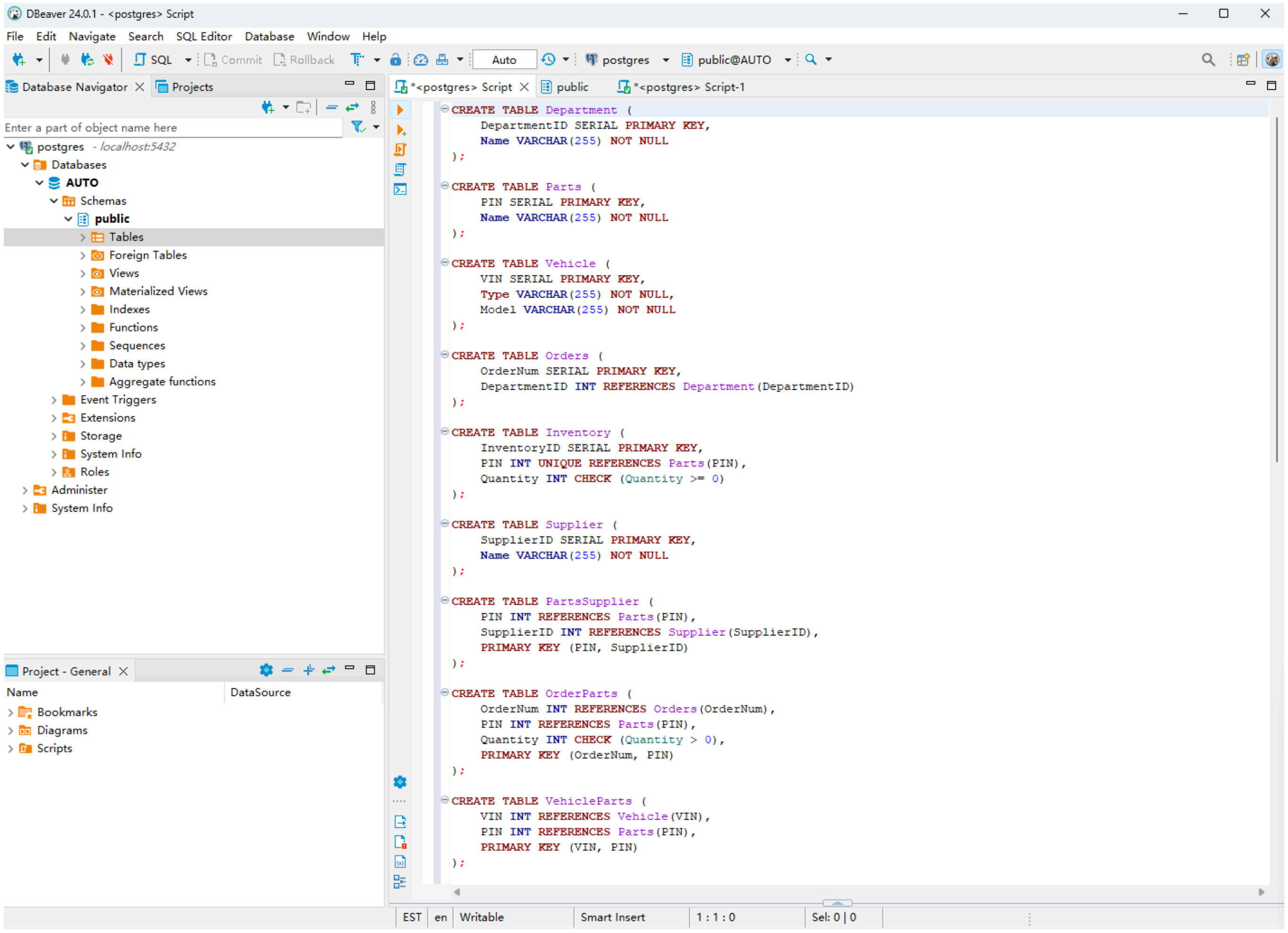
This model maps the previous EER model to a logical data model that shows the references between keys and the cardinality of the entities


VehicleParts
PK VIN int NOT NULL
FK pin int unique not null
Department
PK departmentid int unique NOT NULL
dept_name varchar not null
OrderParts
PK pin int unique NOT NULL
PK ordernum int unique NOT NULL
quantity int NOT NULL
Orders
PK ordernum int unique NOT NULL
FK departmentid int unique NOT NULL
VehicleDepartment
PK departmentid int unique NOT NULL
PK vin int unique NOT NULL
Vehicle
PK vin int unique NOT NULL
Vehicle_type varchar(30) NOT NULL
model varchar(30) NOT NULL
Inventory
PK inventoryid int unique NOT NULL
FK pin int unique NOT NULL
quantity int NOT NULL
Parts
PK pin int unique NOT NULL
partname varchar(30) NOT NULL
Supplier
PK supplierid int unique NOT NULL
suppliername varchar(30) NOT NULL
PartsSupplier
PK pin int unique NOT NULL
FK supplierid int unique NOT NULL

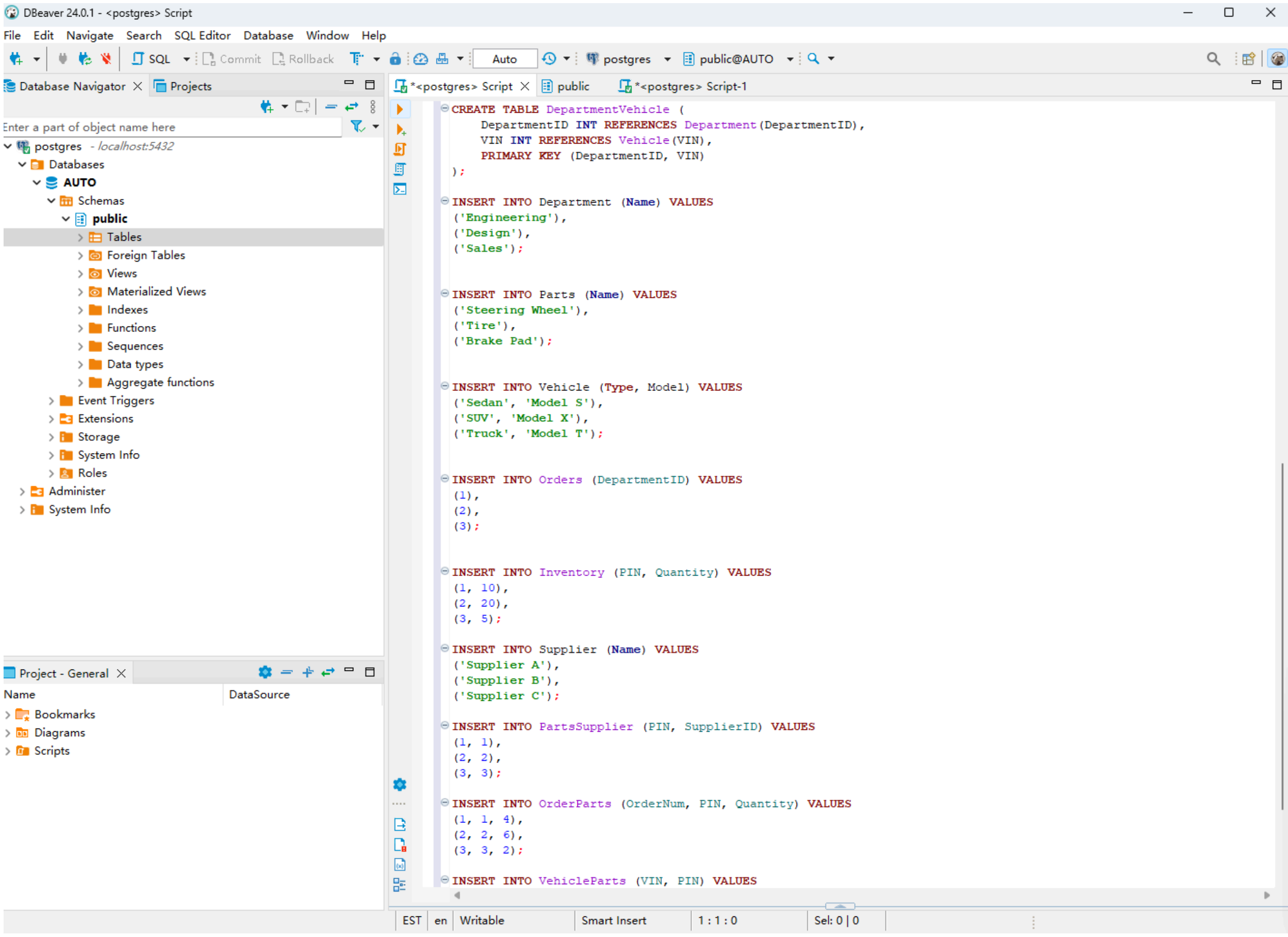
Implementation of the SQL Scripts:

The implementation of the EER model into SQL creates tables that allow for the insertion of data from the many departments (involved) with Automata Inc . Using DDL then, ultimately, DML, tables were created to mirror the relationships between departments depicted in the previously created EER model. The first tables created are **'Department'**, **'Parts',** and **'Supplier'**. They each consist of an identification number that serves as the table's primary key, along with the name of the department and part respectively. The identification number is a serial number, increasing by one each time a new row is added to the table. These tables are important as many of the tables reference them. The '**Vehicle'** table similarly has its own identification number for each row but contains a little more descriptive information, with a column for the vehicle's make as well as the vehicle's model; both implemented as strings with variable length. The **'Orders'** table contains a primary key 'OrderNum', also a serial number, and also a foreign key 'DepartmentID'. The foreign key references the department to which the order belongs to. The **'Inventory'** table has a primary key, 'InventoryID', which is a serial number for identification. The foreign key, 'PIN', which allows the user to see if a part is included in the inventory. The 'Quantity' attribute states how much of the specified part is being included in the inventory. There is a check on this attribute making sure that each part in the inventory has a value greater than 0. Finally, included are four tables that were created as the result of many to many relationships. The tables created are, **'PartsSupplier'**, **'OrderParts'**, **'VehicleParts'**, and **'DepartmentVehicle'**. These junction tables each contain two foreign keys referencing the primary keys of the tables from which they are created. The primary key of these tables is a combination of both of the foreign keys. PartsSupplier, VehicleParts, and DepartmentVehicle all only contain these foreign keys with PartsSupplier having references to 'PIN' and 'SupplierID', OrderParts having references to 'VIN' and 'PIN', and DepartmentVehicle having references to 'DepartmentID' and 'VIN'. Similarly to the other three, 'OrderParts' has references to 'OrderNum' and 'PIN', while also containing an attribute for Quantity which determines the amount of the specified part in the specified order.

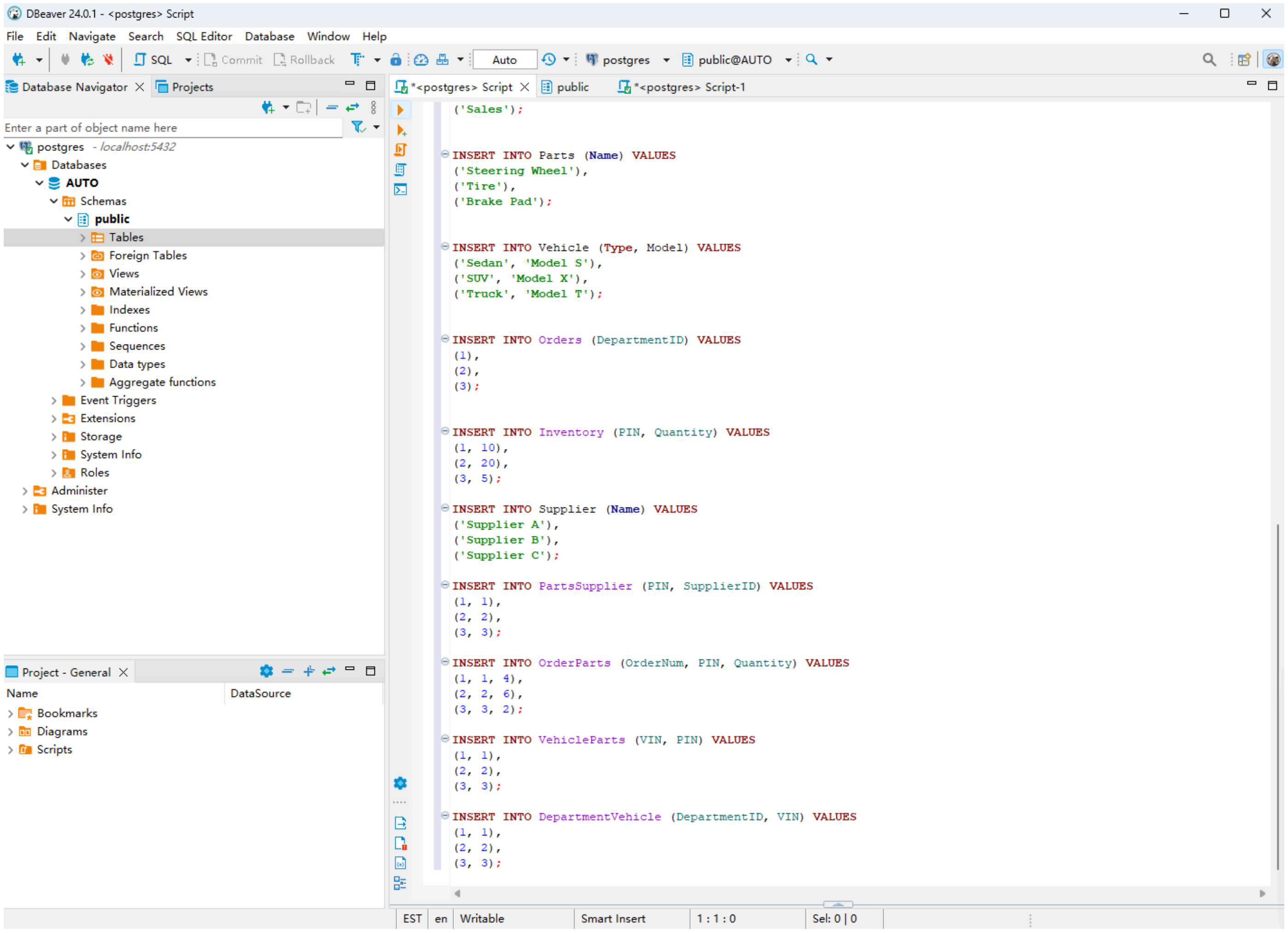
For the insertion of data into these tables, as most of the primary keys are serial numbers that will automatically increase with the creation of a new row, they do not need to be included in the insert statement.



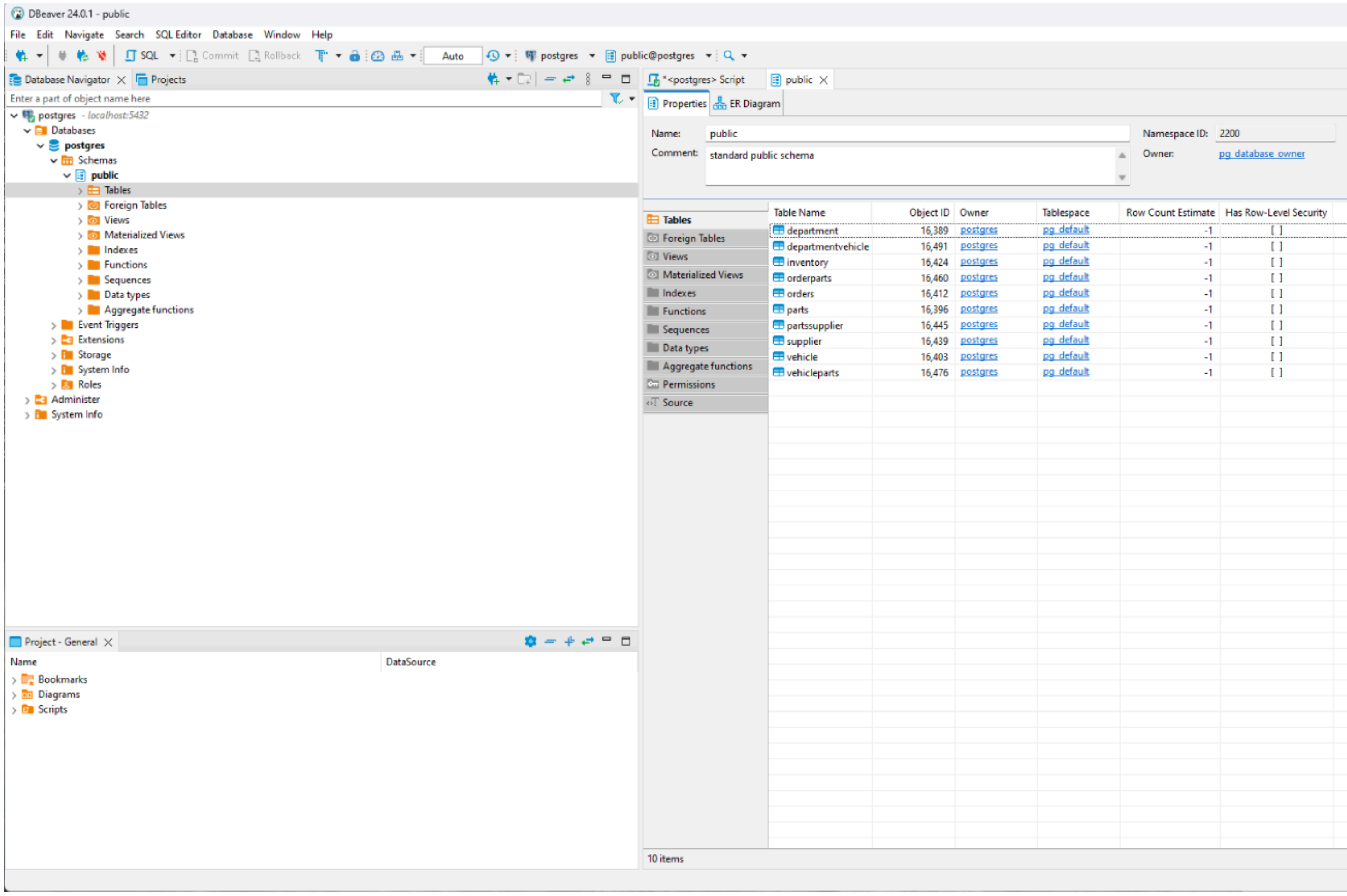
```sql
CREATE TABLE Department (
    DepartmentID SERIAL PRIMARY KEY,
    Name VARCHAR(255) NOT NULL
);

CREATE TABLE Parts (
    PIN SERIAL PRIMARY KEY,
    Name VARCHAR(255) NOT NULL
);

CREATE TABLE Vehicle (
    VIN SERIAL PRIMARY KEY,
    Type VARCHAR(255) NOT NULL,
    Model VARCHAR(255) NOT NULL
);

CREATE TABLE Orders (
    OrderNum SERIAL PRIMARY KEY,
    DepartmentID INT REFERENCES Department(DepartmentID)
);

CREATE TABLE Inventory (
    InventoryID SERIAL PRIMARY KEY,
    PIN INT UNIQUE REFERENCES Parts(PIN),
    Quantity INT CHECK (Quantity >= 0)
);

CREATE TABLE Supplier (
    SupplierID SERIAL PRIMARY KEY,
    Name VARCHAR(255) NOT NULL
);

CREATE TABLE PartsSupplier (
    PIN INT REFERENCES Parts(PIN),
    SupplierID INT REFERENCES Supplier(SupplierID),
    PRIMARY KEY (PIN, SupplierID)
);

CREATE TABLE OrderParts (
    OrderNum INT REFERENCES Orders(OrderNum),
    PIN INT REFERENCES Parts(PIN),
    Quantity INT CHECK (Quantity > 0),
    PRIMARY KEY (OrderNum, PIN)
);

CREATE TABLE VehicleParts (
    VIN INT REFERENCES Vehicle(VIN),
    PIN INT REFERENCES Parts(PIN),
    PRIMARY KEY (VIN, PIN)
);
```


DBeaver 24.0.1 - <postgres> Script
File Edit Navigate Search SQL Editor Database Window Help
SQL Commit Rollback Auto postgres public@AUTO
Database Navigator Projects
*<postgres> Script public *<postgres> Script-1
Enter a part of object name here
postgres - localhost:5432
Databases
AUTO
Schemas
public
Tables
Foreign Tables
Views
Materialized Views
Indexes
Functions
Sequences
Data types
Aggregate functions
Event Triggers
Extensions
Storage
System Info
Roles
Administer
System Info
Project - General
Name DataSource
Bookmarks
Diagrams
Scripts
EST en Writable Smart Insert 1 : 1 : 0 Sel: 0 | 0

```
CREATE TABLE DepartmentVehicle (
    DepartmentID INT REFERENCES Department(DepartmentID),
    VIN INT REFERENCES Vehicle(VIN),
    PRIMARY KEY (DepartmentID, VIN)
);

INSERT INTO Department (Name) VALUES
('Engineering'),
('Design'),
('Sales');


INSERT INTO Parts (Name) VALUES
('Steering Wheel'),
('Tire'),
('Brake Pad');


INSERT INTO Vehicle (Type, Model) VALUES
('Sedan', 'Model S'),
('SUV', 'Model X'),
('Truck', 'Model T');


INSERT INTO Orders (DepartmentID) VALUES
(1),
(2),
(3);


INSERT INTO Inventory (PIN, Quantity) VALUES
(1, 10),
(2, 20),
(3, 5);

INSERT INTO Supplier (Name) VALUES
('Supplier A'),
('Supplier B'),
('Supplier C');

INSERT INTO PartsSupplier (PIN, SupplierID) VALUES
(1, 1),
(2, 2),
(3, 3);

INSERT INTO OrderParts (OrderNum, PIN, Quantity) VALUES
(1, 1, 4),
(2, 2, 6),
(3, 3, 2);

INSERT INTO VehicleParts (VIN, PIN) VALUES
```



```
('Sales');

INSERT INTO Parts (Name) VALUES
('Steering Wheel'),
('Tire'),
('Brake Pad');

INSERT INTO Vehicle (Type, Model) VALUES
('Sedan', 'Model S'),
('SUV', 'Model X'),
('Truck', 'Model T');

INSERT INTO Orders (DepartmentID) VALUES
(1),
(2),
(3);

INSERT INTO Inventory (PIN, Quantity) VALUES
(1, 10),
(2, 20),
(3, 5);

INSERT INTO Supplier (Name) VALUES
('Supplier A'),
('Supplier B'),
('Supplier C');

INSERT INTO PartsSupplier (PIN, SupplierID) VALUES
(1, 1),
(2, 2),
(3, 3);

INSERT INTO OrderParts (OrderNum, PIN, Quantity) VALUES
(1, 1, 4),
(2, 2, 6),
(3, 3, 2);

INSERT INTO VehicleParts (VIN, PIN) VALUES
(1, 1),
(2, 2),
(3, 3);

INSERT INTO DepartmentVehicle (DepartmentID, VIN) VALUES
(1, 1),
(2, 2),
(3, 3);
```



Name: public
Comment: standard public schema
Namespace ID: 2200
Owner: pg_database_owner

Table Name	Object ID	Owner	Tablespace	Row Count Estimate	Has Row-Level Security
department	16,389	postgres	pg_default	-1	[ ]
departmentvehicle	16,491	postgres	pg_default	-1	[ ]
inventory	16,424	postgres	pg_default	-1	[ ]
orderparts	16,460	postgres	pg_default	-1	[ ]
orders	16,412	postgres	pg_default	-1	[ ]
parts	16,396	postgres	pg_default	-1	[ ]
partssupplier	16,445	postgres	pg_default	-1	[ ]
supplier	16,439	postgres	pg_default	-1	[ ]
vehicle	16,403	postgres	pg_default	-1	[ ]
vehicleparts	16,476	postgres	pg_default	-1	[ ]

PK & FK Table

This table aims to catalog the essential elements of the database schema, specifically focusing on the identification and relational mechanisms that underpin the database's integrity and relational logic.

Table name	Primary Key (Composite*)	Foreign Key
Department	DepartmentID	/
Parts	PIN	/

Vehicle	`VIN`	/
Orders	`OrderNum`	`DepartmentID` references `Department(DepartmentID)`
Inventory	`InventoryID`	`PIN` references `Parts(PIN)` and is unique, ensuring a one-to-one relationship with the `Parts` table.
Supplier	`SupplierID`	/

PartsSupplier	`(PIN, SupplierID)`	`PIN` references `Parts(PIN)`
		
		`SupplierID` references `Supplier(SupplierID)`

OrderParts	`(OrderNum, PIN)`	`OrderNum` references `Orders(OrderNum)`
		
		`PIN` references `Parts(PIN)`

VehicleParts	`(VIN, PIN)`	`VIN` references `Vehicle(VIN)`
		
		`PIN` references `Parts(PIN)`

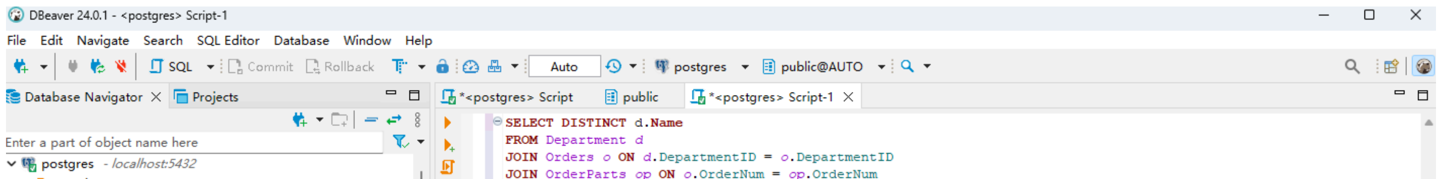
DepartmentVehicle	`(DepartmentID, VIN)`	`DepartmentID` references `Department(DepartmentID)`
		
		`VIN` references `Vehicle(VIN)`

<u>The Implementation of the 6 Typical Queries in “PostgreSQL”</u>

The queries allow users to ask questions about the database and the six created queries are sample examples to show how the results can be found based on the questions asked. The first question is “Which departments have ordered a 'Steering Wheel'?” The question seeks to identify all departments within AUtomata, Inc. that have placed orders for “Steering Wheels”. To retrieve this information, a query is crafted to join the ‘Department’, ‘Order’, and ‘OrderItem’ tables, along with the ‘Item’ table where the item’s name matches ‘Steering Wheel’. The method involves using SQL ‘JOIN’ operations to correlate the relevant tables and a ‘WHERE’ clause to filter for the specific item. The second question is “What are the current inventory levels for each part?”. The objective here is to report on the quantity of each item or part currently available in inventory. This requires accessing data from the ‘Inventory’ and ‘Item’ tables. The approach is straightforward, employing a ‘JOIN’ between these tables to associate each part with its current stock level, presenting a clear picture of inventory status. The third question is “List all parts supplied by 'Supplier A'.”. This question aims to enumerate all items or parts that are provided by a specific supplier, named ‘Supplier A’. By utilizing the ‘ItemSupplier’ join table, which maps items to suppliers, in conjunction with the ‘Supplier’ and ‘Item’ tables, a query can filter by the supplier’s name and list all associated parts. This involves ‘JOIN’ operations and a ‘WHERE’ clause targeting the supplier’s name. The fourth question is “Which supplier has been used most frequently for orders placed by the “Engineering” department?”. This inquiry focuses on identifying the supplier with the highest frequency of orders originating from the Engineering department. The solution involves complex SQL queries that aggregate order data from the ‘Orders’, ‘OrderItem’, and ‘ItemSupplier’ tables, filtered by the Engineering department, and count the occurrence of each supplier, sorting the result to find the top supplier. The fifth question is “How many orders have been placed by each department?”.The goal here is to quantify the number of orders initiated by each department. This is achieved through aggregating data within the ‘Orders’ table based on the department identifier, using a ‘GROUP BY’ clause on the department and a ‘COUNT’ function to tally orders per department. This query provides insights into the ordering patterns and demands of different departments. The last question is “What is the total number of parts ordered for trucks?”. This question requires calculating the cumulative quantity of parts ordered specifically for truck production. It necessitates linking orders to vehicle types through potentially a **‘VehicleParts’**and **‘OrderItem’** relationship, then filtering for 'Trucks' and summing up the quantities of parts ordered. This complex query likely involves multiple **‘JOIN’** operations and a summation aggregation to arrive at the total parts ordered.

Question 1

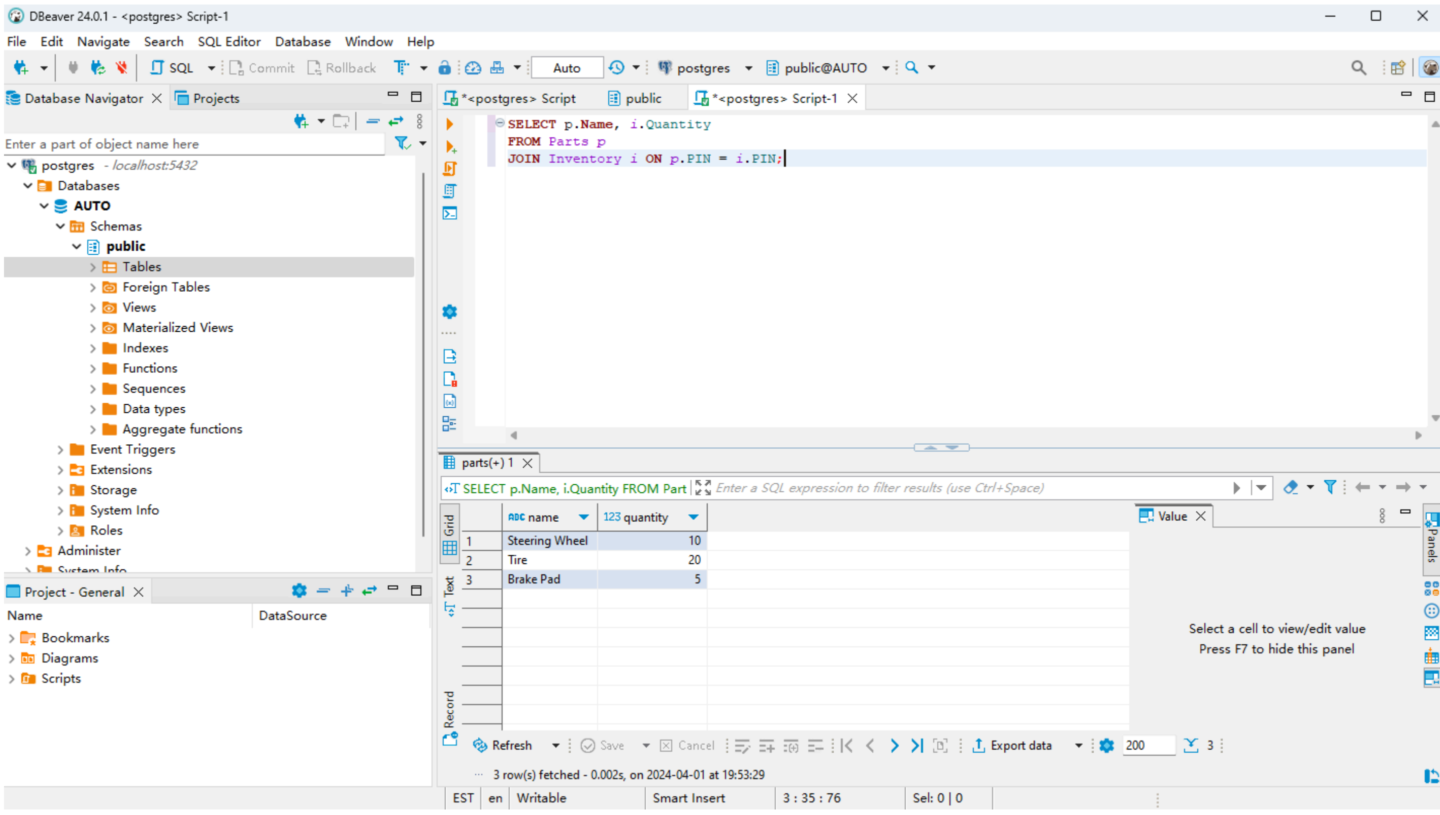
Question: Which departments have ordered a 'Steering Wheel'?



```
SELECT DISTINCT d.Name
FROM Department d
JOIN Orders o ON d.DepartmentID = o.DepartmentID
JOIN OrderParts op ON o.OrderNum = op.OrderNum
```

Question 2

Question: What are the current inventory levels for each part?



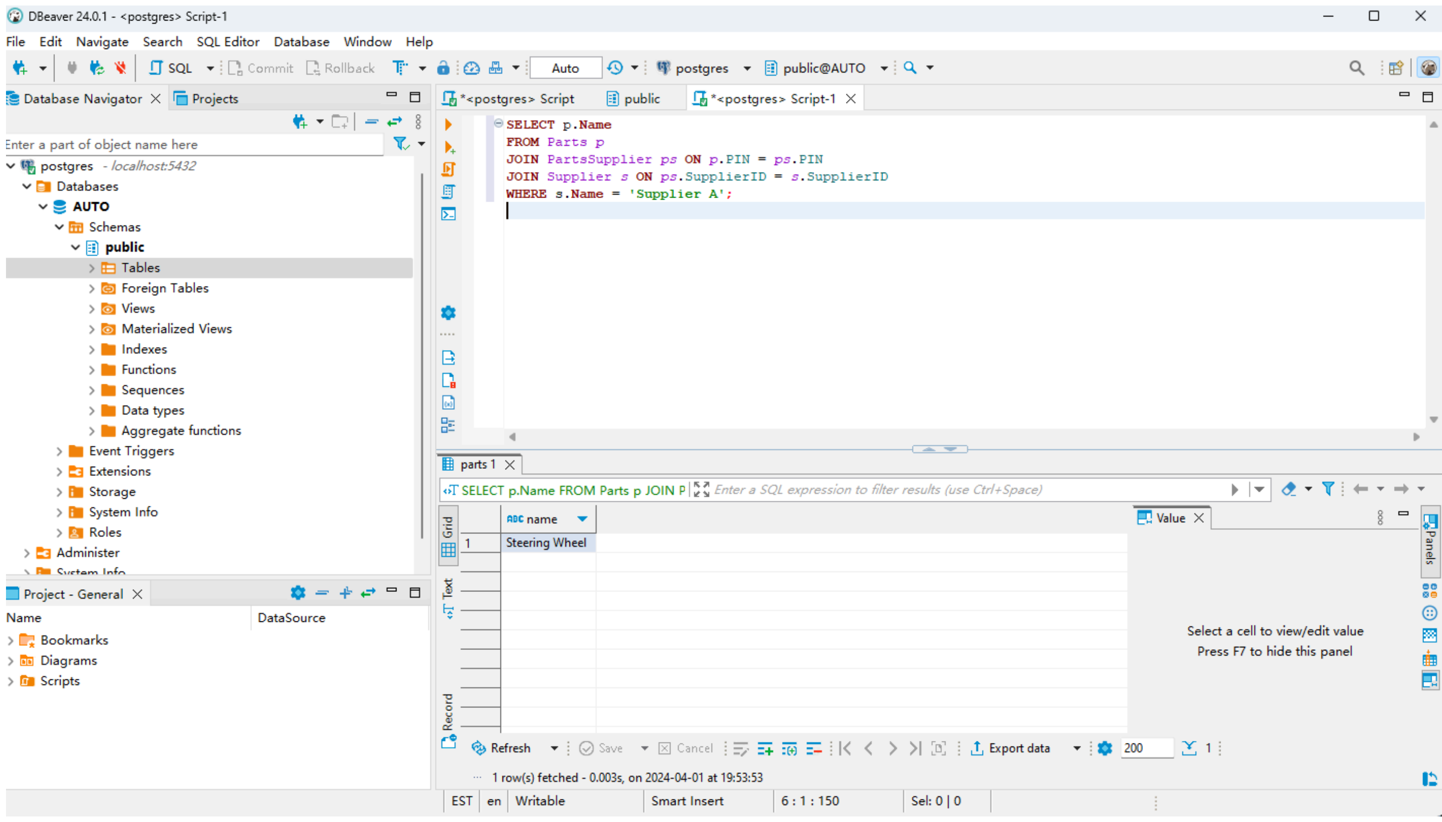
```
SELECT p.Name, i.Quantity
FROM Parts p
JOIN Inventory i ON p.PIN = i.PIN;
```

	name	quantity
1	Steering Wheel	10
2	Tire	20
3	Brake Pad	5

3 row(s) fetched - 0.002s, on 2024-04-01 at 19:53:29

Question 3

Question: List all parts supplied by 'Supplier A'.



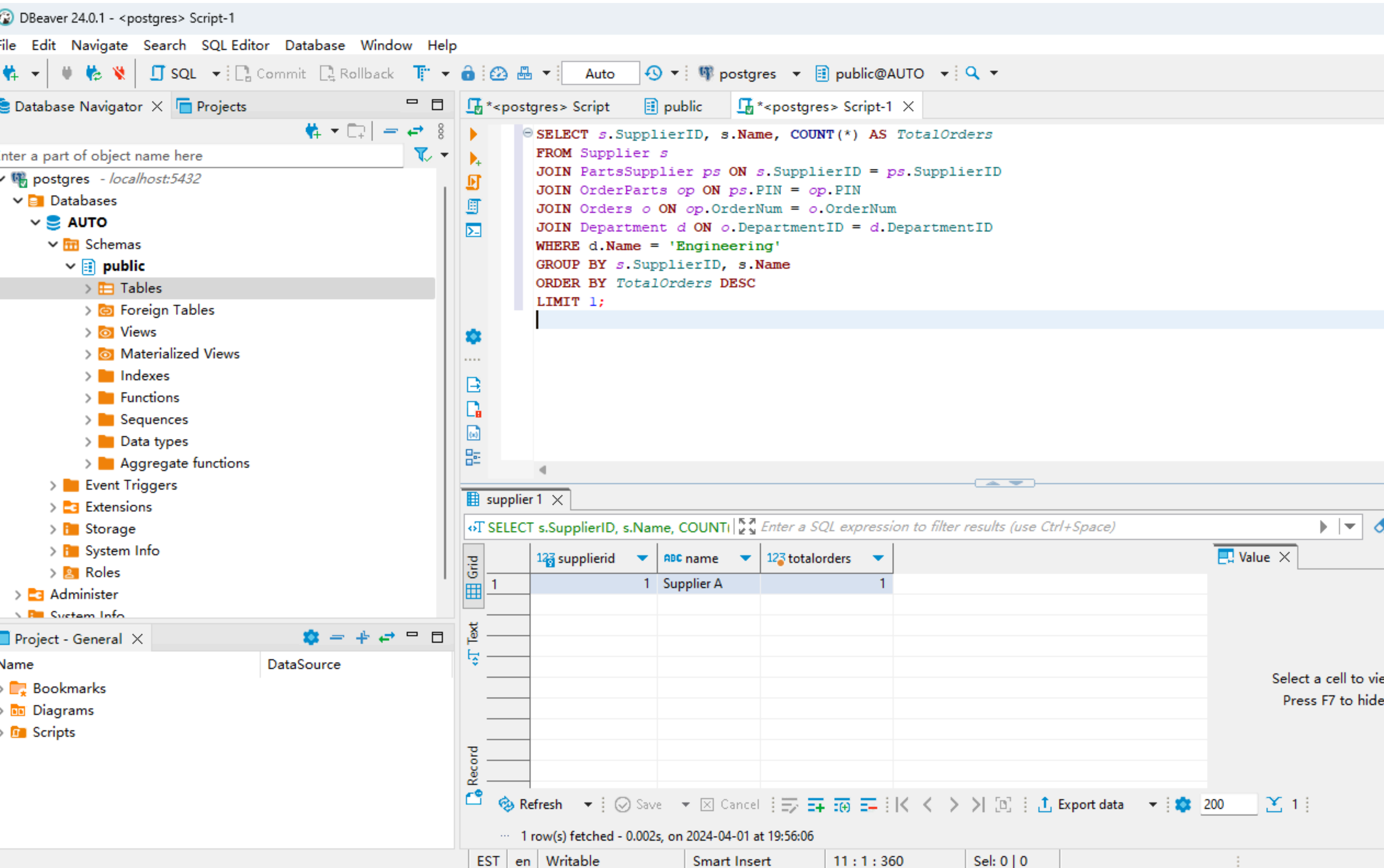
```
SELECT p.Name
FROM Parts p
JOIN PartsSupplier ps ON p.PIN = ps.PIN
JOIN Supplier s ON ps.SupplierID = s.SupplierID
WHERE s.Name = 'Supplier A';
```

	name
1	Steering Wheel

```
1 row(s) fetched - 0.003s, on 2024-04-01 at 19:53:53
```

Question 4

Question: Which supplier has been used most frequently for orders placed by the “Engineering” department?



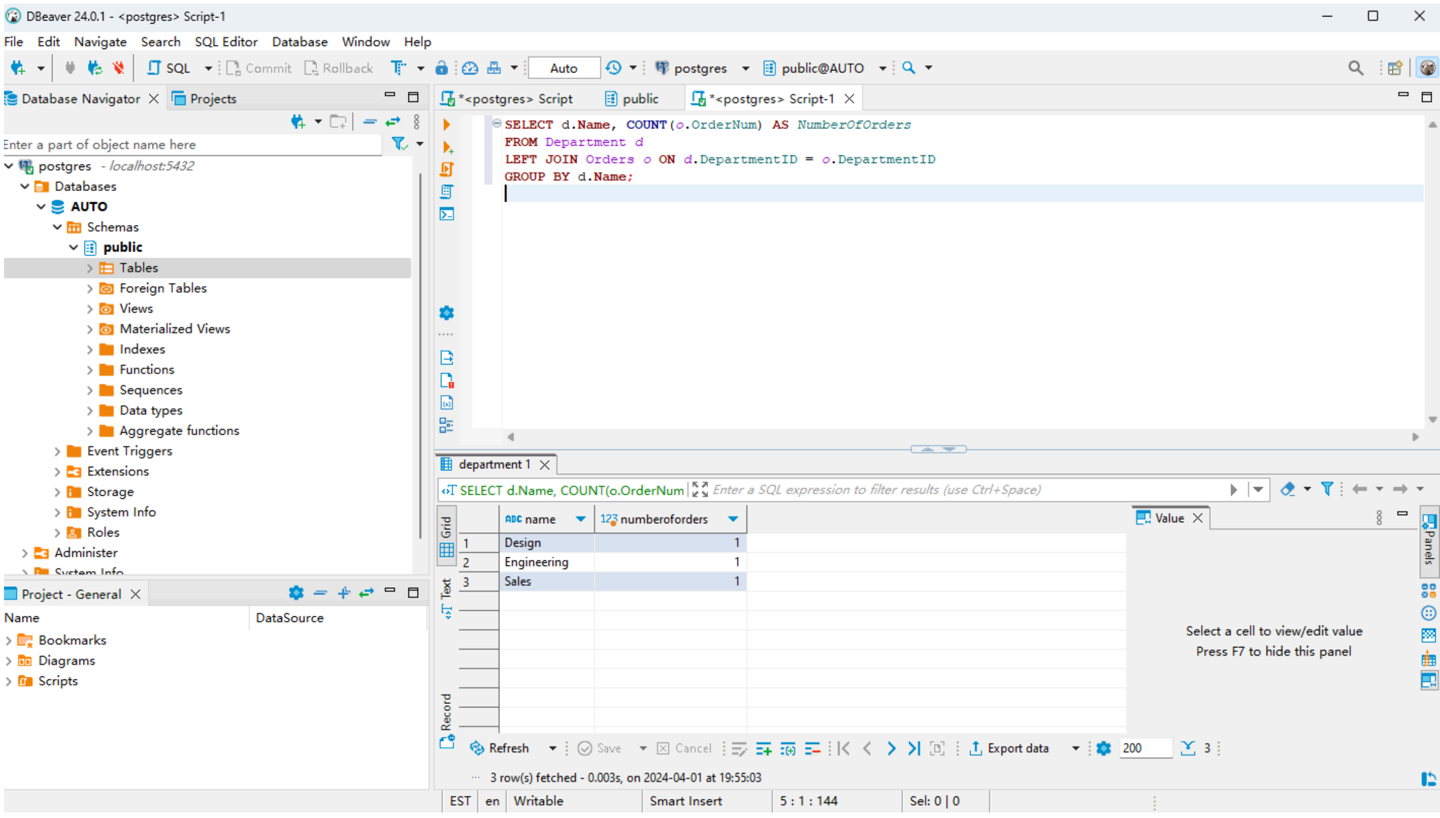
```sql
SELECT s.SupplierID, s.Name, COUNT(*) AS TotalOrders
FROM Supplier s
JOIN PartsSupplier ps ON s.SupplierID = ps.SupplierID
JOIN OrderParts op ON ps.PIN = op.PIN
JOIN Orders o ON op.OrderNum = o.OrderNum
JOIN Department d ON o.DepartmentID = d.DepartmentID
WHERE d.Name = 'Engineering'
GROUP BY s.SupplierID, s.Name
ORDER BY TotalOrders DESC
LIMIT 1;
```

	supplierid	name	totalorders
1	1	Supplier A	1

1 row(s) fetched - 0.002s, on 2024-04-01 at 19:56:06

Question 5

Question: How many orders have been placed by each department?



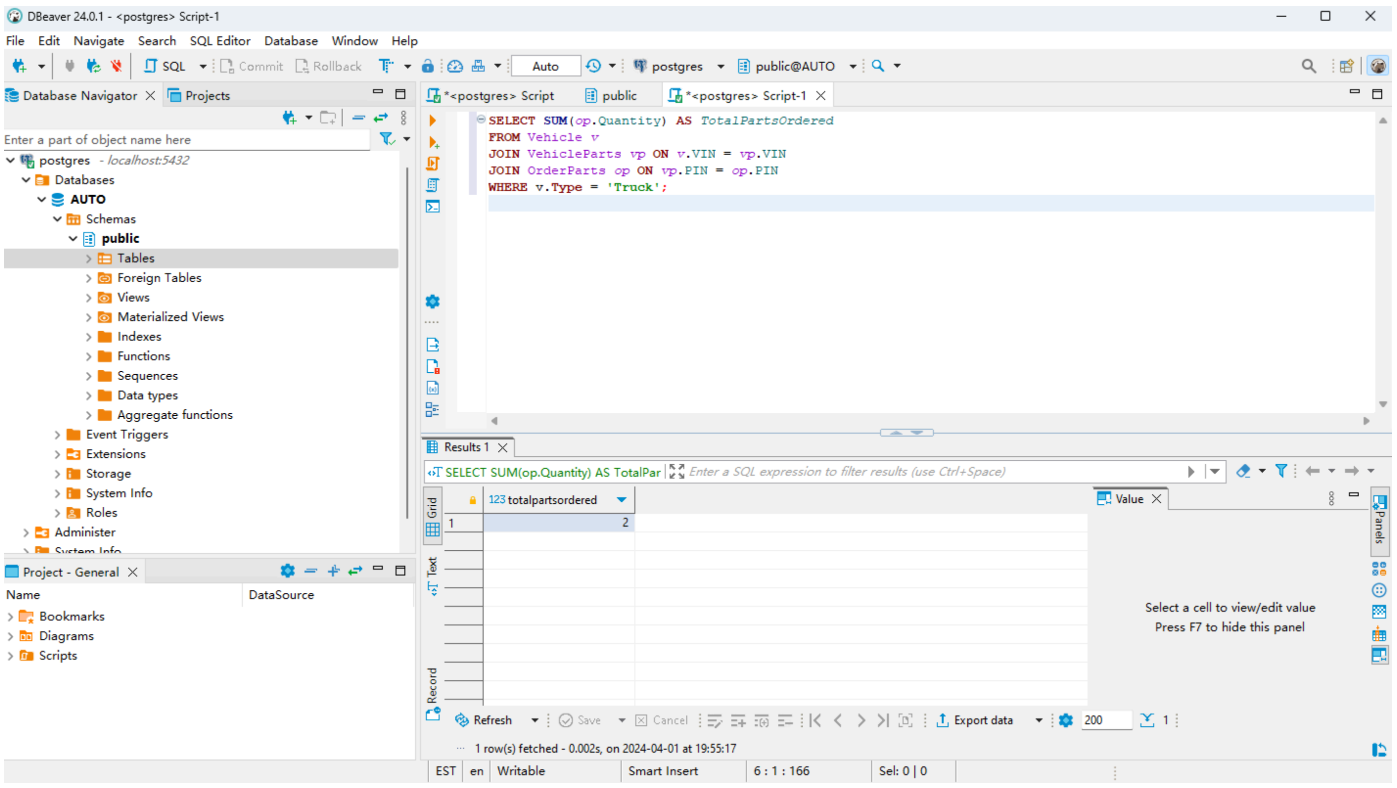
```
SELECT d.Name, COUNT(o.OrderNum) AS NumberOfOrders
FROM Department d
LEFT JOIN Orders o ON d.DepartmentID = o.DepartmentID
GROUP BY d.Name;
```

	name	numberoforders
1	Design	1
2	Engineering	1
3	Sales	1

3 row(s) fetched - 0.003s, on 2024-04-01 at 19:55:03

Question 6

Question: What is the total number of parts ordered for trucks?



```
SELECT SUM(op.Quantity) AS TotalPartsOrdered
FROM Vehicle v
JOIN VehicleParts vp ON v.VIN = vp.VIN
JOIN OrderParts op ON vp.PIN = op.PIN
WHERE v.Type = 'Truck';
```

```
   totalpartsordered
1                  2
```

1 row(s) fetched - 0.002s, on 2024-04-01 at 19:55:17

Conclusion:

The biggest challenge our group faced was communication and expectation. As the project manager, I am solely responsible for this part of the project. I admit I had trouble fitting into this role. At first, it was difficult to find common times that we could all meet because we all have vastly different schedules that made it hard to accommodate. Also due to my early insistence on conducting meetings in-person, there were a handful of times when these meetings only included two members. So, we did not efficiently use our time in the beginning stage of this project.

As dynamic as a project can be, it is extremely important to have a system of communication in place that can allow individuals to work together at various hours of the day, and document that progress. Once we had an established MS Teams channel, group communication really started to improve. Improvements for the next project would be to implement MS teams in the very beginning because our project would certainly have been finished sooner. Also, zoom meetings would allow for extra flexibility that is necessary with team collaboration, and will be a crucial component of our next group project. During this assignment, members got sick and could not attend classes which only reduced the awareness of not only the individual but the overall group. But if we had used Zoom or Facetime as a meeting location, there would not have been that confusion.

Our objective was to, as a team, create a database for the company Automata with SQL implementing all methods of DB development that we learned in class starting with an EER Model, writing those models into SQL scripts, and then writing queries that pertain to the questions that workers for Automata would be interested in finding. All within the time constraint given and to do it as efficiently as possible.