

A repair order is used to track a repair. Each repair order has a number that is generated by the system, an origination date, and a completion date. A repair order has any number of notes that are used to describe the complaints, as well as document the course of a repair. A repair order is for a vehicle. After the car is diagnosed, the mechanic will decide what procedures are needed to repair it. The available procedures are stored in the information system as procedure definitions.

In some cases, a repair estimate is prepared. In other cases, a repair is done directly. In any case, an invoice of line items required for the repair is prepared with a description, quantity, and price for each line item. A procedure may have many line items and a line item pertains to a specific procedure. The invoice is printed either as an estimate or as a final bill when the repair is complete and payable. There can be multiple invoices per repair order, including partial billing or pre-payment, as well as multiple estimates.

Identification of the relations (entity types):

Entity name	Description	Aliases	Occurrence
RepairOrder	General term describing all repair order information.	Repair Order	Each repair order is specific to one particular vehicle. A particular vehicle can have many repair orders over time
Procedure	General term describing all procedures that can be done on a vehicle.	Service	A particular procedure can be used on many different repair orders, but doesn't have to be used on any. A repair order has at least one, but possibly many different procedures.
LineItem	General term describing all purchases (parts or labor) needed to complete a procedure.	Line Item	Any given line item can be present on 0 or more (possibly many) invoices. A single invoice contains at least one, but possibly many line items. A procedure may have many line items and a line item pertains to a specific procedure
LineItemPart	Specialization of LineItem	Parts	A Part is a type of line item
LineItemLabor	Specialization of LineItem	Labor	Labor is a type of line item
Invoice	General term describing the collection of all line items for a transaction.	Invoice, Bill	Each invoice is specific to one repair order. A repair order can have multiple invoices.
InvoiceEstimate	Specialization of Invoice, an	Estimate	An Estimate is a type of invoice.

	estimated cost of repairs.		
InvoiceCost	Specialization of Invoice, a payable bill.	Cost	A Cost is a type of invoice.
Vehicle	General term describing a vehicle being repaired.	Vehicle, Automobile, Car	Each vehicle can have many repair orders, a repair order is specific to one vehicle.
Mechanic	General term describing the staff member choosing the procedures.	Staff	Each mechanic can choose many procedures or none, any given procedure can be chosen by many mechanics or none. A mechanic can fulfill many repair orders or none, each repair order must be fulfilled by exactly one mechanic.

Identification of relationship types and their participation and cardinality constraints:

Entity Name	Multiplicity	Relationship	Multiplicity	Entity Name
RepairOrder	0..*	Fixes	1..1	Vehicle
	0..*	Documents	1..*	Procedure
	1..1	Generates	1..*	Invoice
Procedure	0..*	Requires	1..*	LineItem
Invoice	0..*	Lists	1..*	LineItem
Mechanic	0..*	Chooses	0..*	Procedure
	1..1	Fulfils	0..*	RepairOrder

Logical Data Model

Identification of attributes and association of attributes with entity or relationship types:

Entity or relationship name	Attributes	Description	Data Type & Length	Nulls	Multivalued
RepairOrder	repairNo (PK)	Uniquely identifies repair orders	VARCHAR 10	No	No
	originDate	Date when invoice is created.	DATE 10	No	No
	completeDate	Date when repair completes.	DATE 10	No	No
	complaintNote	Document containing complaints about vehicle.	TEXT	No	No
	vehicleNo (FK)	Identity of vehicle receiving repairs	VARCHAR 10	No	No
	staffNo (FK)	Identity of mechanic(s) in charge of repairs	VARCHAR 10	No	No
Procedure	procedureNo (PK)	Uniquely identifies procedures	VARCHAR 10	No	No
	procedureDescription	Gives a description of procedure	TEXT	No	No
LineItem	itemNo (PK)	Uniquely identifies a line item	VARCHAR 10	No	No
	itemName	Name of line item	VARCHAR 50	No	No
	description	Describes the line item	TEXT	No	No
	procedureNo (FK)	Identify the procedure the line item is used for.	VARCHAR 10	No	No
				No	No
LineItemPart	price	Gives the price of the line item part.	DECIMAL (7,2)	No	No
LineItemLabor	rate	Gives hourly labor rate	DECIMAL (5,2)	No	No
	hours	Gives number of hours required for labor	DECIMAL (5,2)	No	No
			DECIMAL (7,2)	No	No

	price (derived as rate * hours)	Gives the price of the labor, derived from hourly rate and number hours.			
Invoice	invoiceNo (PK) odometerStart invoicePrintDate repairNo (FK)	Uniquely identifies an invoice. Odometer reading when repairs start. Date that the invoice is printed Reference to RepairOrder the invoice belongs to.	VARCHAR 10 INTEGER INTEGER VARCHAR 10	No No No No	No No No No
InvoiceEstimate					
InvoiceCost	odometerFinish paymentDate	Odometer reading when repairs finish. Date the invoice is paid for.	INTEGER DATE	No Yes	No No
Vehicle	vin (PK) make year model	Vehicle vin number Make of the vehicle Year of the vehicle Model of the vehicle	VARCHAR 17 VARCHAR 50 INTEGER VARCHAR 50	No No No No	No No No No
Mechanic	staffNo (PK)	Uniquely identifies a mechanic	VARCHAR 10	No	No
Lists	quantity	Indicates the quantity of a particular line item in the relationship Invoice Lists Lineltem.	INTEGER	Yes	No
Documents	repairDocument	Document outlining the repair process.	TEXT	No	No

Determination of candidate and primary key attributes of entity types:

Entity	Primary Key	Candidate Keys
RepairOrder	repairNo	N/A
Procedure	procedureNo	N/A
LineItem	itemNo	itemName
Invoice	invoiceNo	N/A
Vehicle	vin	N/A
Mechanic	staffNo	N/A

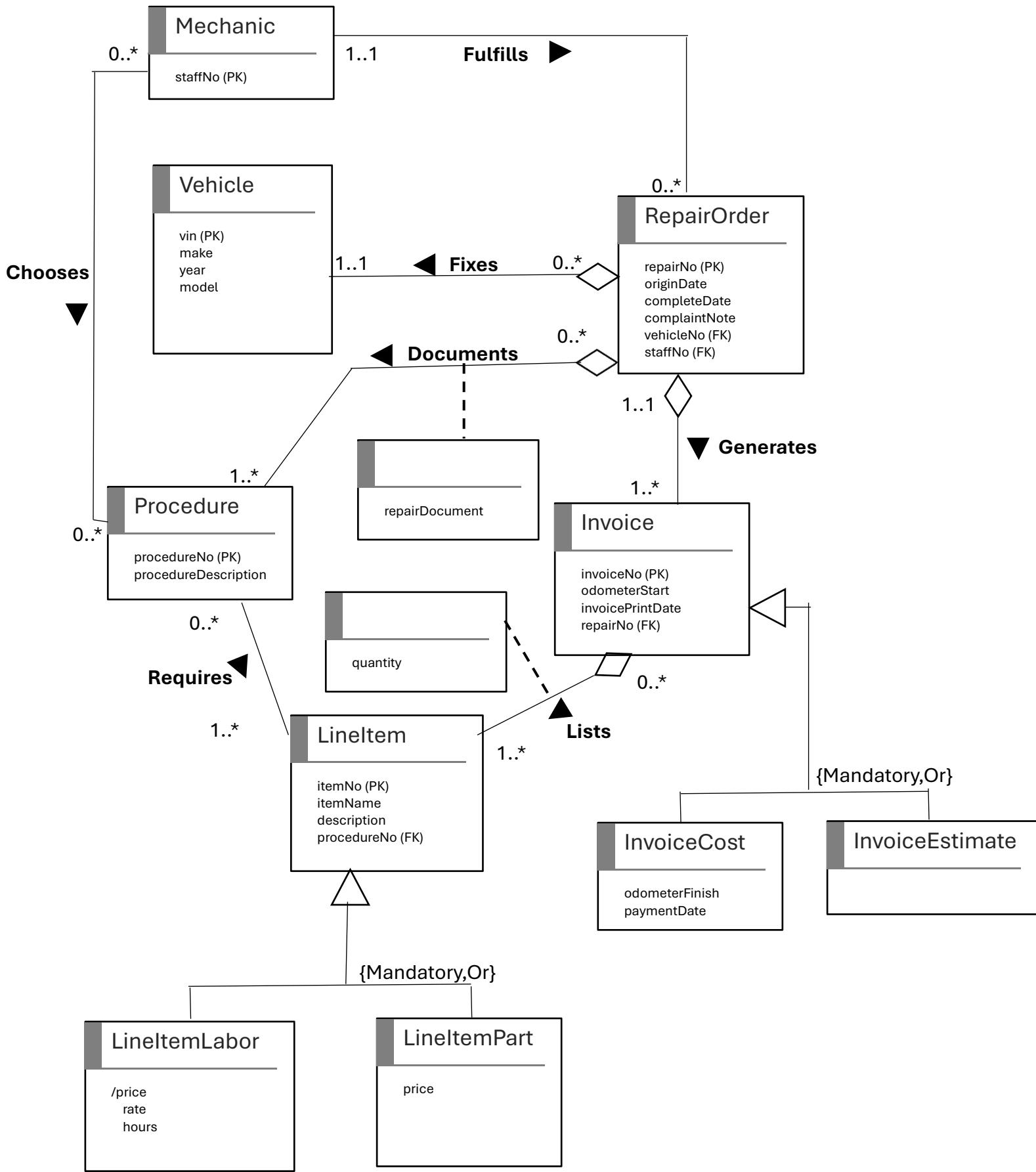
Determination of specialization/generalization and categorization relationships:

Specialization/Generalization Relationship:

- **Invoice (Mandatory and Disjoint)**
 - **Estimate**
 - **Final Bill**
- **Line Item (Mandatory and Disjoint)**
 - **Parts Line Item**
 - **Labor Line Item**

Aggregation Relationship:

- **Repair Order**
 - **Invoice**
 - **Procedure**
 - **Vehicle**
- **Invoice**
 - **Line Item**



Conceptual Data Model

Derivation of relations from the refined conceptual model:

Strong entity types:

RepairOrder(repairNo, originDate, completeDate, complaintNote)

Primary key: repairNo

Procedure(procedureNo, procedureDescription)

Primary key: procedureNo

LineItem(itemNo, itemName, description)

Primary key: itemNo

Alternate key: itemName

Invoice(invoiceNo, odometerStart, invoicePrintDate)

Primary key: invoiceNo

Vehicle(vin, make, year, model) **Changed from conceptual model, vin is unique and more easily accessed as primary key than vehicleNo.**

Primary key: vin

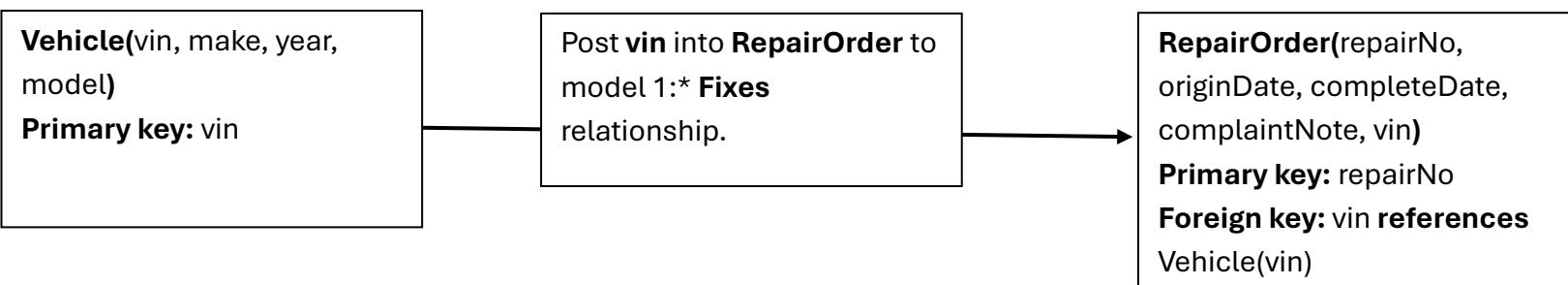
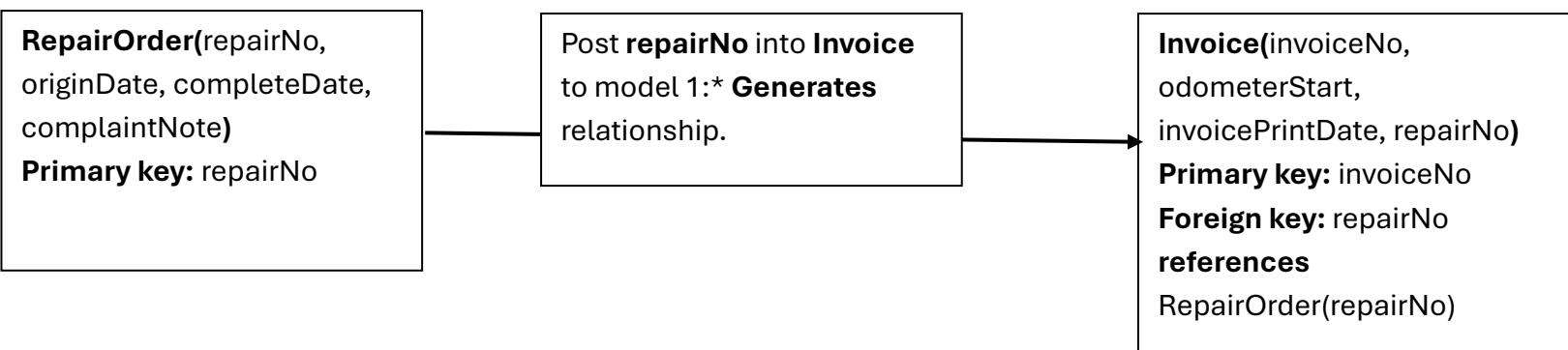
Mechanic(staffNo, fName, lName)

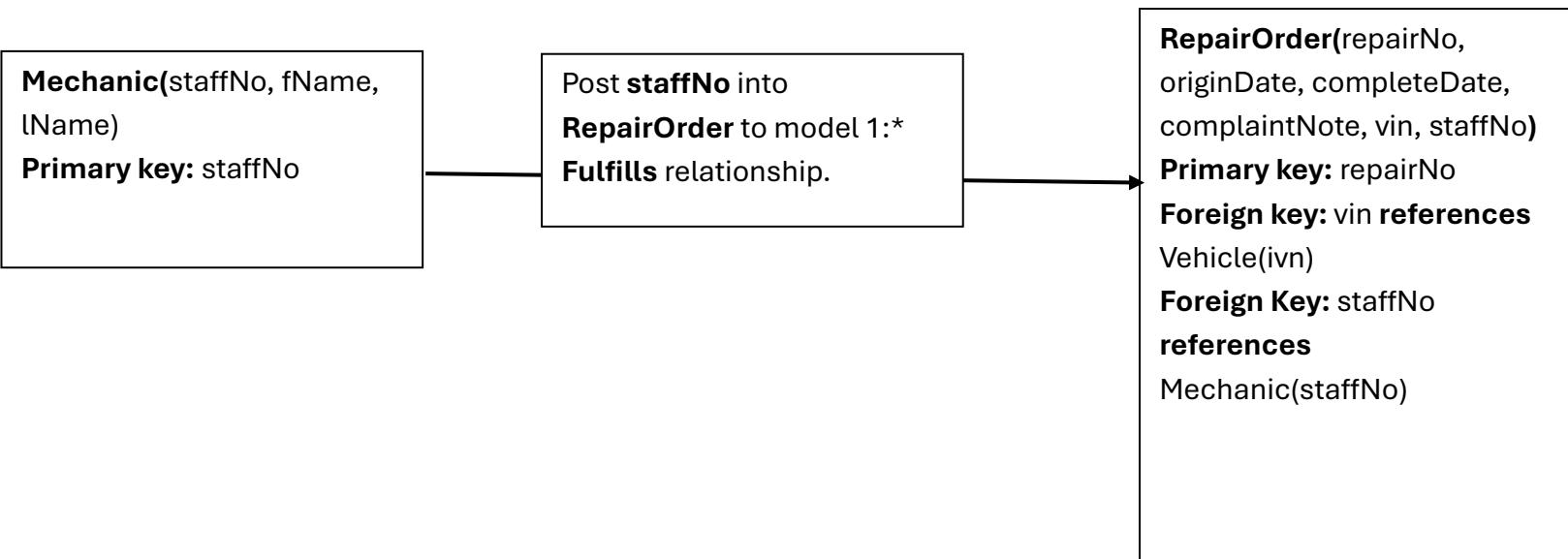
Primary key: staffNo

Weak Entity Types:

None

One-to-many (1:*) binary relationship types:





One-to-one (1:1) binary relationship types:

- a) Mandatory participation on both sides:

None

- b) Mandatory Participation on one side:

None

- c) Optional participation on both sides:

None

One-to-one recursive relationships:

None

Superclass/Subclass Relationship Types:

Mandatory, Disjoint:

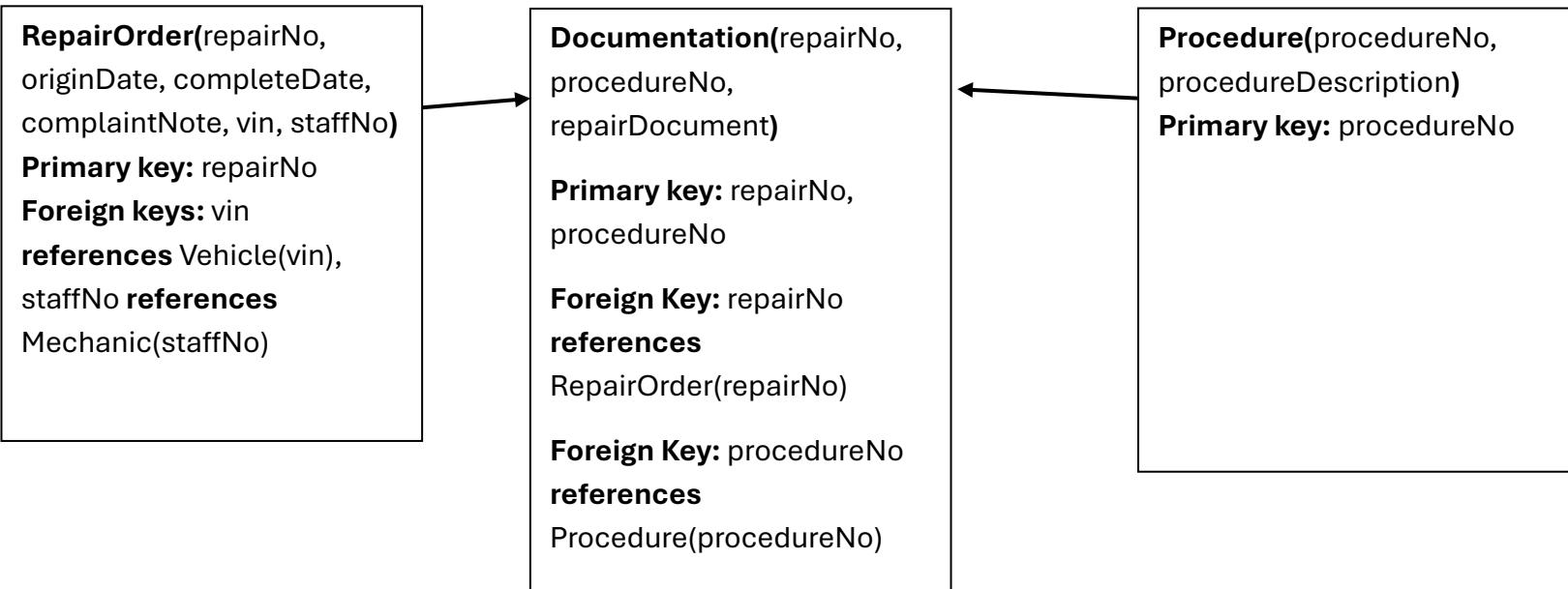
LineItemLabor(itemNo, rate, hours, itemName, description)

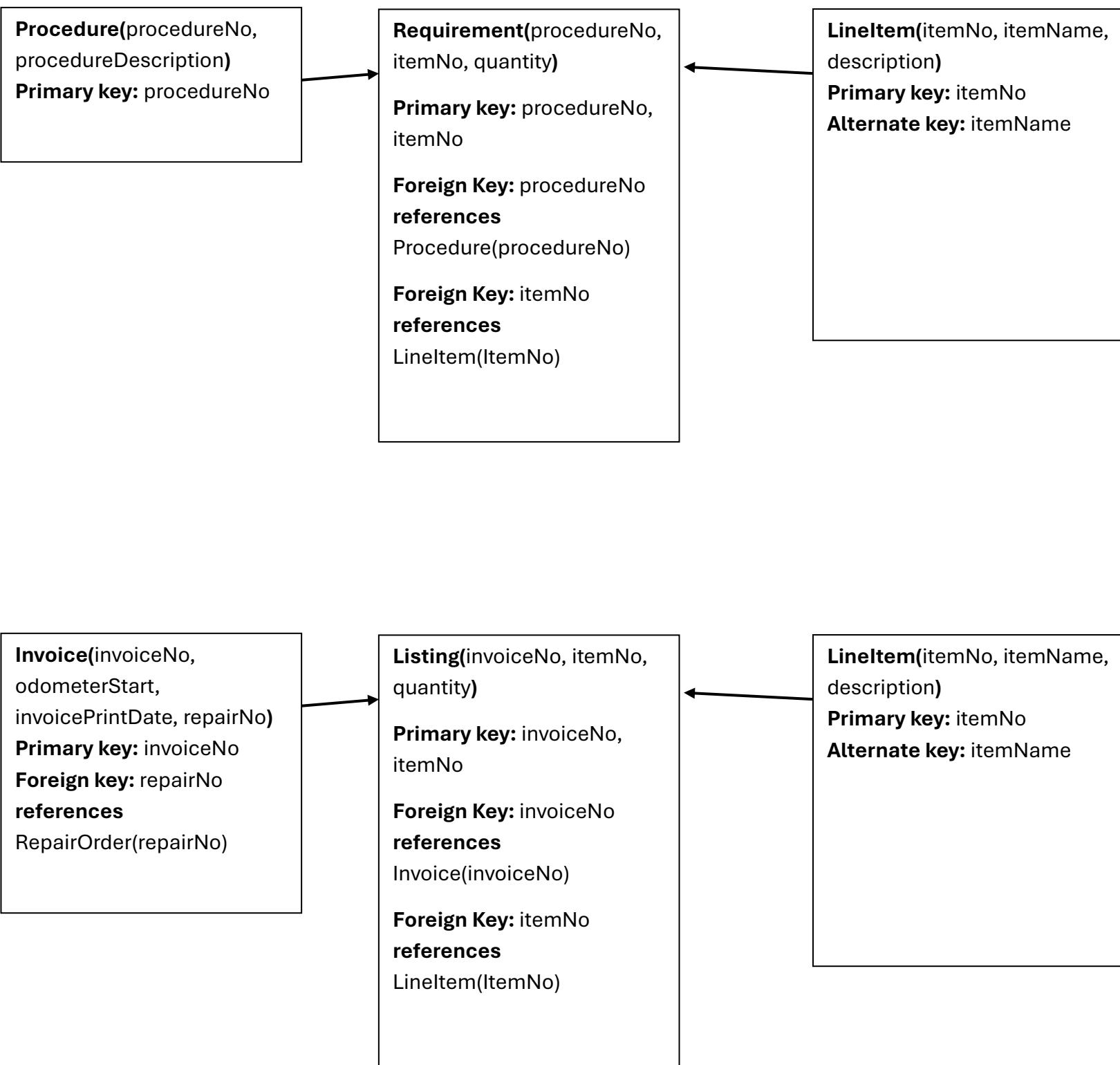
Primary key: itemNo

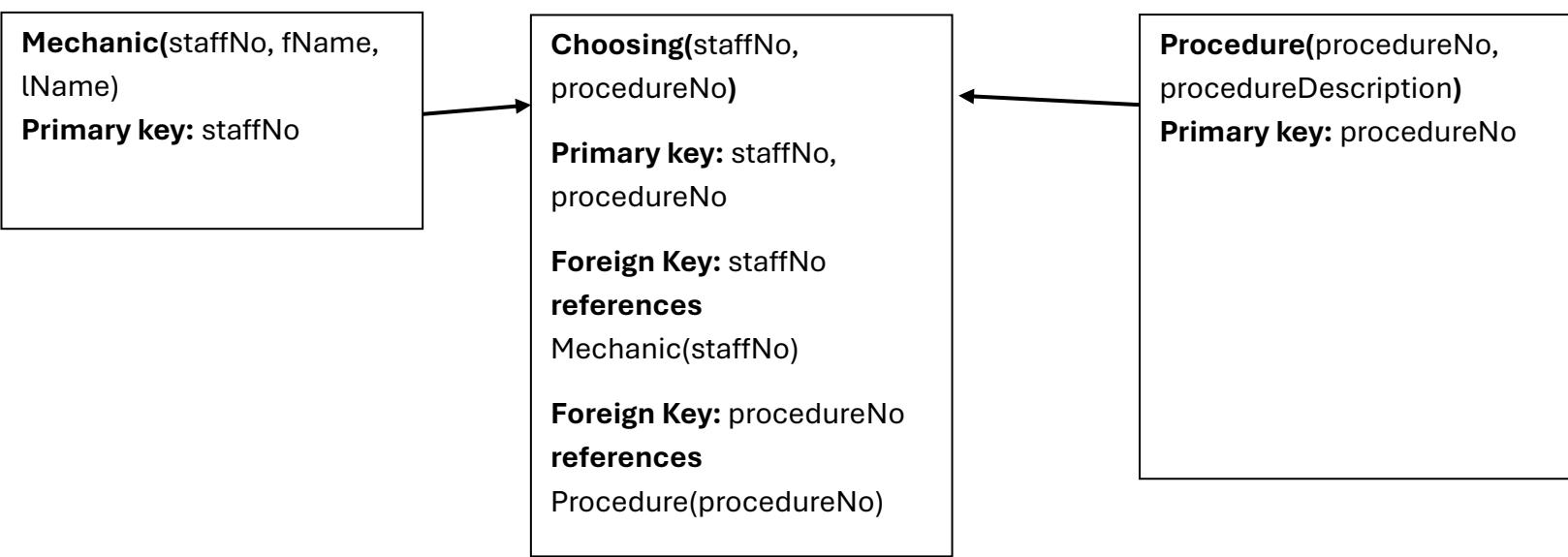
LineItemPart(itemNo, price, itemName, description)

Primary key: itemNo

Many-to-many (*:*) binary relationship types:







Complex relationship types:

None

Relations:

RepairOrder (repairNo, originDate, completeDate, complaintNote, vin, staffNo) Primary key: repairNo Foreign key: vin references Vehicle(vin) Foreign Key: staffNo references Mechanic(staffNo)	Documentation (repairNo, procedureNo, repairDocument) Primary key: repairNo, procedureNo Foreign Key: repairNo references RepairOrder(repairNo) Foreign Key: procedureNo references Procedure(procedureNo)
Procedure (procedureNo, procedureDescription) Primary key: procedureNo	Requirement (procedureNo, itemNo, quantity) Primary key: procedureNo, itemNo Foreign Key: procedureNo references Procedure(procedureNo) Foreign Key: itemNo references LineItem(itemNo)

LineItem (itemNo, itemName, description) Primary key: itemNo Alternate key: itemName	Listing (invoiceNo, itemNo, quantity) Primary key: invoiceNo, itemNo Foreign Key: invoiceNo references Invoice(invoiceNo) Foreign Key: itemNo references LineItem(itemNo)
Invoice (invoiceNo, odometerStart, invoicePrintDate, repairNo) Primary key: invoiceNo Foreign key: repairNo references RepairOrder(repairNo)	Choice (staffNo, procedureNo) Primary key: staffNo, procedureNo Foreign Key: staffNo references Mechanic(staffNo) Foreign Key: procedureNo references Procedure(procedureNo)
Vehicle (vin, make, year, model) Primary key: vin	
Mechanic (staffNo, fName, lName) Primary key: staffNo	

Multi-valued attributes:

None

Validate relations using normalization:

Repair Order:

repairNo -> originDate, completeDate, complaintNote, vin, staffNo

1NF: No repeating groups

2NF: Every non-primary-key attribute is fully functionally dependent on primary key

repairNo (no partial functional dependency).

3NF: No non-primary-key attribute is transitively dependent on the primary key **repairNo**.

BCNF: Determinant is candidate key (primary key)

Procedure

procedureNo -> procedureDescription

1NF: No repeating groups

2NF: The non-primary-key attribute is functionally dependent on **procedureNo** (no partial functional dependency).

3NF: The non-primary-key attribute is transitively dependent on the primary key **procedureNo**.

BCNF: Determinant is candidate key (primary key)

Linelitem

itemNo → itemName, description

1NF: No repeating groups

2NF: Every non-primary-key attribute is fully functionally dependent on primary key **itemNo** (no partial functional dependency)

3NF: No non-primary-key attribute is transitively dependent on the primary key **itemNo**.

BCNF: Determinant is candidate key (primary key)

Invoice

invoiceNo → odometerStart, invoicePrintDate, repairNo

1NF: No repeating groups

2NF: Every non-primary-key attribute is fully functionally dependent on primary key **invoiceNo** (no partial functional dependency). Primary key is not composite.

3NF: No non-primary-key attribute is transitively dependent on the primary key **invoiceNo**.

BCNF: Determinant is candidate key (primary key)

Vehicle

vin → make, year, model

1NF: No repeating groups

2NF: Every non-primary-key attribute is fully functionally dependent on primary key **vin** (no partial functional dependency). Primary key is not composite.

3NF: No non-primary-key attribute is transitively dependent on the primary key **vin**.

BCNF: Determinant is candidate key (primary key).

Mechanic

staffNo \rightarrow fName, lName

1NF: No repeating groups

2NF: Every non-primary-key attribute is fully functionally dependent on primary key

staffNo (no partial functional dependency). Primary key is not composite

3NF: No non-primary-key attribute is transitively dependent on the primary key **staffNo**.

BCNF: Determinant is candidate key (primary key)

Documentation

repairNo, procedureNo \rightarrow repairDocument

1NF: No repeating groups

2NF: Every non-primary-key attribute is fully functionally dependent on primary key

repairNo, procedureNo (no partial functional dependency). **repairDocument** is specific to the combination of **repairNo** and **procedureNo**.

3NF: The non-primary-key attribute is not transitively dependent on the primary key **repairNo, procedureNo**.

BCNF: Determinant is candidate key (primary key)

Requirement

procedureNo, itemNo \rightarrow quantity

1NF: No repeating groups

2NF: Every non-primary-key attribute is fully functionally dependent on primary key **procedureNo, itemNo** (no partial functional dependency). **quantity** is specific to the combination of **procedureNo** and **itemNo**.

3NF: The non-primary-key attribute is not transitively dependent on the primary key **procedureNo, itemNo**.

BCNF: Determinant is candidate key (primary key)

Listing

invoiceNo, itemNo → quantity

1NF: No repeating groups

2NF: Every non-primary-key attribute is fully functionally dependent on primary key **invoiceNo, itemNo** (no partial functional dependency). **quantity** is specific to the combination of **invoiceNo** and **itemNo**.

3NF: The non-primary-key attribute is not transitively dependent on the primary key **invoiceNo, itemNo**.

BCNF: Determinant is candidate key (primary key)

Choosing

staffNo, procedureNo

1NF: No repeating groups

2NF: There is nothing to depend on. No non-primary attributes. No partial functional dependency.

3NF: No non-primary-key attributes, no transitive dependency.

BCNF: No determinant, just primary key.

Validation of logical model against corresponding user transactions:

Repair Order:

A **Repair Order** is used to track a repair. Each repair order has a number that is generated by the system, an origination date, and a completion date. A repair order has any number of notes that are used to describe the complaints, as well as document the course of a repair. A repair order is for a **Vehicle**.

Mechanic:

After the car is diagnosed, the **Mechanic** will decide what **Procedures** are needed to repair it.

Procedure:

The available procedures are stored in the information system as procedure definitions. Some procedure definitions include: Replacing starter motor; Replacing accelerator cable; State inspection; Lube, oil, and oil filter; Replacing tail light bulb; Replacing head light bucket; Repairing and replacing Pitman arm; and Servicing and repairing transmission.

Invoice:

In some cases, a repair estimate is prepared. In other cases, a repair is done directly. In any case, an **Invoice of Line Items** required for the repair is prepared with a description, quantity, and price for each line item. The invoice is printed either as an **estimate** or as a **final bill** when the repair is complete and payable. There can be multiple invoices per repair order, including partial billing or pre-payment, as well as multiple estimates. The invoice includes the following information: Invoice number, which is listed on the document handed to the customer. Odometer mileage when the vehicle comes in and when it is finished. Date when the repair originated (from the repair order). Date when the repair is complete (from the repair order). Date when the invoice is printed. Date paid.

Line Item:

A **Procedure** may have many line items and a **Line Item** pertains to a specific procedure. Examples of line items include: Starter; Labor for starter replacement; Towing; Accelerator cable; Labor for replacing the accelerator cable; Bulb; Labor for lube, oil, and oil filter; Oil filter; Headlight adjusting screw; and State safety inspection.

Definition of integrity constraints:

Primary Key Constraints/ Entity Integrity:

RepairOrder

Primary Key Constraints/ Entity Integrity:

Primary key: repairNo must be unique and not null.

Foreign Key Constraints/Referential Integrity

Foreign key: vin **references** Vehicle(vin) ON UPDATE CASCADE ON DELETE CASCADE

Foreign Key: staffNo **references** Mechanic(staffNo) ON UPDATE CASCADE ON DELETE NO ACTION

To Insert a staffNo into the **RepairOrder** relation, it must already exist in the **Mechanic** relation.

Procedure

Primary Key Constraints/ Entity Integrity:

Primary key: procedureNo must be unique and not null.

LineItem

Primary Key Constraints/ Entity Integrity:

Primary key: itemNo must be unique and not null.

Invoice

Primary Key Constraints/ Entity Integrity:

Primary key: invoiceNo must be unique and not null.

Foreign Key Constraints/Referential Integrity

Foreign key: repairNo **references** RepairOrder(repairNo) ON UPDATE CASCADE ON DELETE CASCADE

Vehicle

Primary Key Constraints/ Entity Integrity:

Primary key: vin must be unique and not null.

Mechanic

Primary Key Constraints/ Entity Integrity:

Primary key: staffNo must be unique and not null.

Documentation

Primary Key Constraints/ Entity Integrity:

Primary key: repairNo, procedureNo must be unique and not null.

Foreign Key Constraints/Referential Integrity

Foreign Key: repairNo **references** RepairOrder(repairNo) ON UPDATE CASCADE ON DELETE CASCADE

Foreign Key: procedureNo **references** Procedure(procedureNo) ON UPDATE CASCADE ON DELETE NO ACTION

Requir

Primary Key Constraints/ Entity Integrity:

Primary key: procedureNo, itemNo must be unique and not null.

Foreign Key Constraints/Referential Integrity

Foreign Key: procedureNo **references** Procedure(procedureNo) ON UPDATE CASCADE ON DELETE CASCADE

Foreign Key: itemNo **references** LineItem(itemNo) ON UPDATE CASCADE ON DELETE NO ACTION

Listing

Primary Key Constraints/ Entity Integrity:

Primary key: invoiceNo, itemNo must be unique and not null.

Foreign Key Constraints/Referential Integrity

Foreign Key: invoiceNo **references** Invoice(invoiceNo) ON UPDATE CASCADE ON DELETE CASCADE

Foreign Key: itemNo **references** LineItem(itemNo) ON UPDATE CASCADE ON DELETE NO ACTION

Choosing

Primary Key Constraints/ Entity Integrity:

Primary key: staffNo, procedureNo must be unique and not null.

Foreign Key Constraints/Referential Integrity

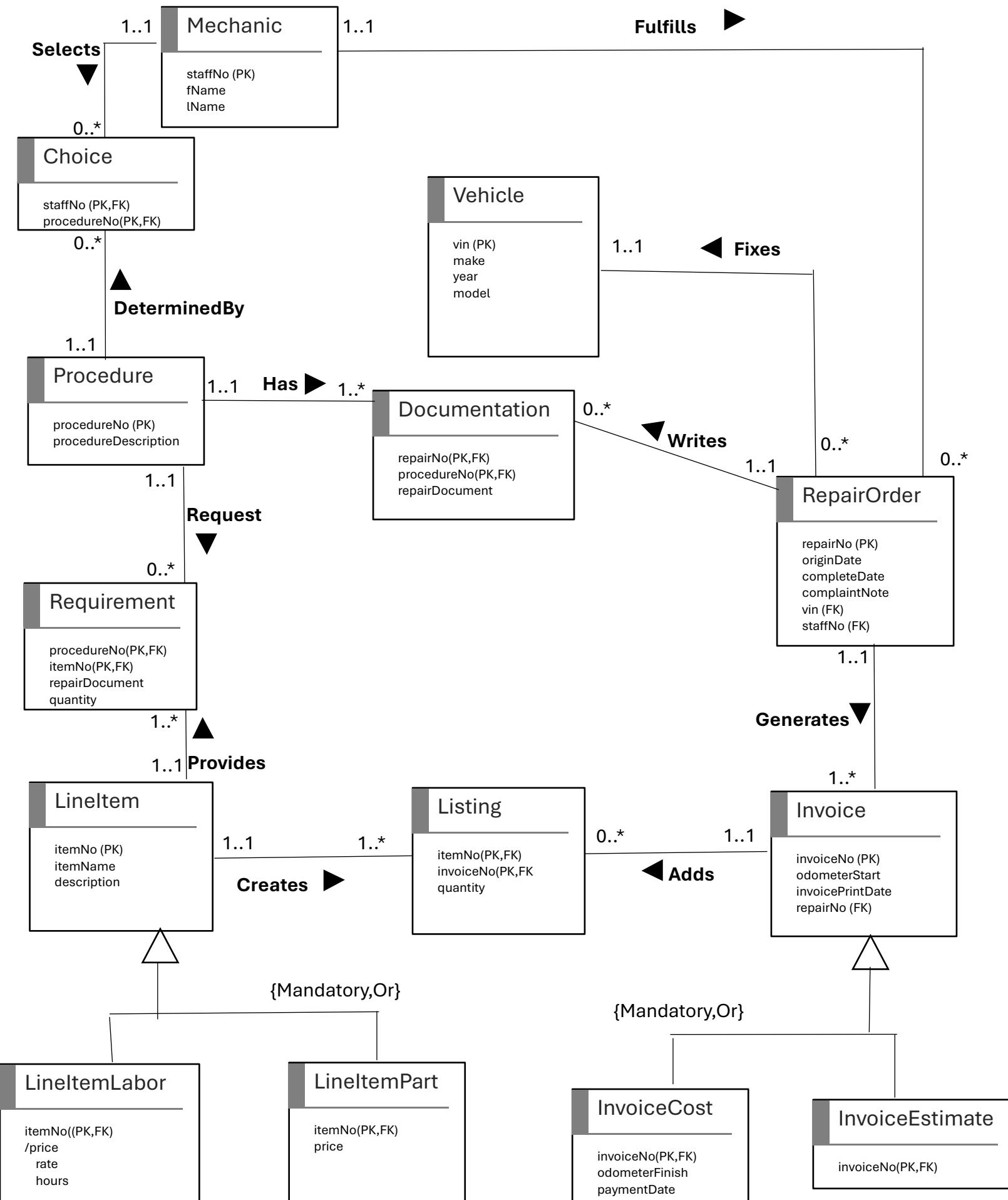
Foreign Key: staffNo **references** Mechanic(staffNo) ON UPDATE CASCADE ON DELETE CASCADE

Foreign Key: procedureNo **references** Procedure(procedureNo) ON UPDATE CASCADE ON DELETE NO ACTION

General Constraints:

General constraints could be added, but there are none specified in the instructions. For example, there could be constraints implemented limiting the number of Repair Orders a single mechanic could have active at one time.

UPDATED EER DIAGRAM for Conceptual Model:

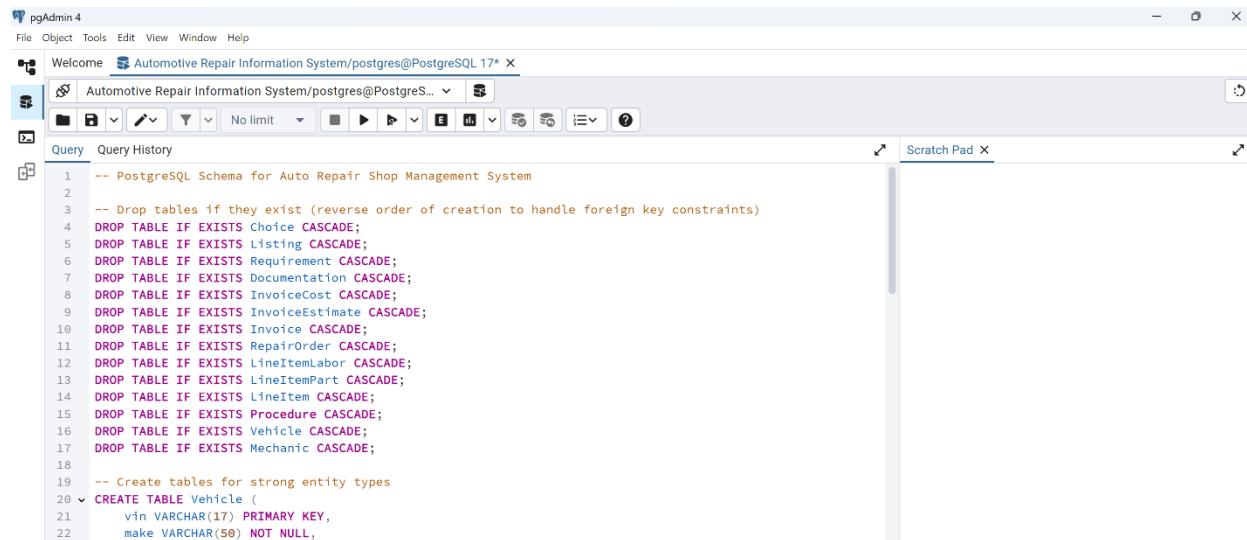


SQL Statements and screenshots:

Table Creation

-- PostgreSQL Schema for Auto Repair Shop Management System

```
-- Drop tables if they exist (reverse order of creation to handle foreign key constraints)
DROP TABLE IF EXISTS Choice CASCADE;
DROP TABLE IF EXISTS Listing CASCADE;
DROP TABLE IF EXISTS Requirement CASCADE;
DROP TABLE IF EXISTS Documentation CASCADE;
DROP TABLE IF EXISTS InvoiceCost CASCADE;
DROP TABLE IF EXISTS InvoiceEstimate CASCADE;
DROP TABLE IF EXISTS Invoice CASCADE;
DROP TABLE IF EXISTS RepairOrder CASCADE;
DROP TABLE IF EXISTS LineItemLabor CASCADE;
DROP TABLE IF EXISTS LineItemPart CASCADE;
DROP TABLE IF EXISTS LineItem CASCADE;
DROP TABLE IF EXISTS Procedure CASCADE;
DROP TABLE IF EXISTS Vehicle CASCADE;
DROP TABLE IF EXISTS Mechanic CASCADE;
```



The screenshot shows the pgAdmin 4 interface. The title bar says "pgAdmin 4". The main window has a toolbar at the top with various icons. Below the toolbar is a menu bar with "File", "Object", "Tools", "Edit", "View", "Window", and "Help". The main area is divided into two panes: "Query" on the left and "Scratch Pad" on the right. The "Query" pane contains the SQL code for the schema. The "Scratch Pad" pane is currently empty.

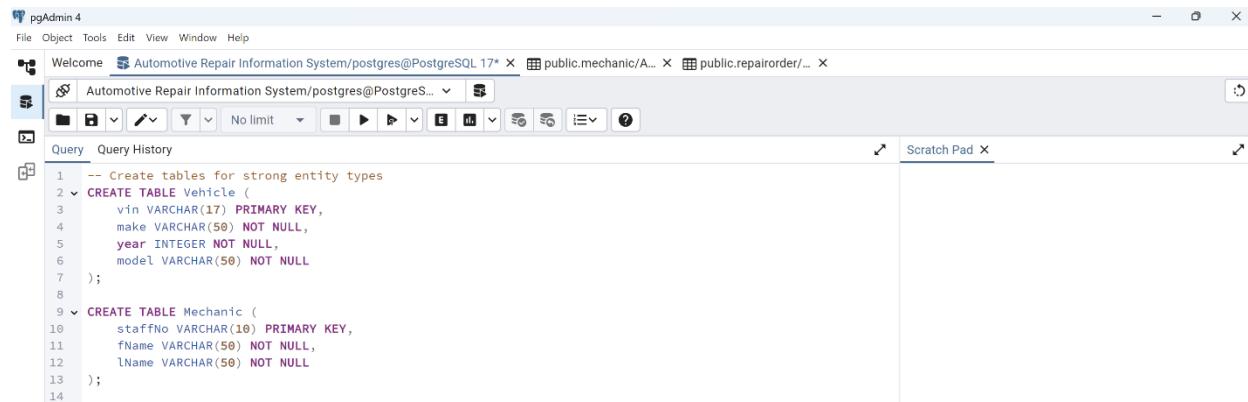
```
1 -- PostgreSQL Schema for Auto Repair Shop Management System
2
3 -- Drop tables if they exist (reverse order of creation to handle foreign key constraints)
4 DROP TABLE IF EXISTS Choice CASCADE;
5 DROP TABLE IF EXISTS Listing CASCADE;
6 DROP TABLE IF EXISTS Requirement CASCADE;
7 DROP TABLE IF EXISTS Documentation CASCADE;
8 DROP TABLE IF EXISTS InvoiceCost CASCADE;
9 DROP TABLE IF EXISTS InvoiceEstimate CASCADE;
10 DROP TABLE IF EXISTS Invoice CASCADE;
11 DROP TABLE IF EXISTS RepairOrder CASCADE;
12 DROP TABLE IF EXISTS LineItemLabor CASCADE;
13 DROP TABLE IF EXISTS LineItemPart CASCADE;
14 DROP TABLE IF EXISTS LineItem CASCADE;
15 DROP TABLE IF EXISTS Procedure CASCADE;
16 DROP TABLE IF EXISTS Vehicle CASCADE;
17 DROP TABLE IF EXISTS Mechanic CASCADE;
18
19 -- Create tables for strong entity types
20 CREATE TABLE Vehicle (
21     vin VARCHAR(17) PRIMARY KEY,
22     make VARCHAR(50) NOT NULL,
```

-- Create tables for strong entity types

```
CREATE TABLE Vehicle (
    vin VARCHAR(17) PRIMARY KEY,
    make VARCHAR(50) NOT NULL,
```

```
year INTEGER NOT NULL,  
model VARCHAR(50) NOT NULL  
);
```

```
CREATE TABLE Mechanic (  
    staffNo VARCHAR(10) PRIMARY KEY,  
    fName VARCHAR(50) NOT NULL,  
    lName VARCHAR(50) NOT NULL  
);
```



The screenshot shows the pgAdmin 4 interface with a query editor window. The query window contains the following SQL code:

```
1 -- Create tables for strong entity types  
2 v CREATE TABLE Vehicle (  
3     vin VARCHAR(17) PRIMARY KEY,  
4     make VARCHAR(50) NOT NULL,  
5     year INTEGER NOT NULL,  
6     model VARCHAR(50) NOT NULL  
7 );  
8  
9 v CREATE TABLE Mechanic (  
10    staffNo VARCHAR(10) PRIMARY KEY,  
11    fName VARCHAR(50) NOT NULL,  
12    lName VARCHAR(50) NOT NULL  
13 );  
14
```

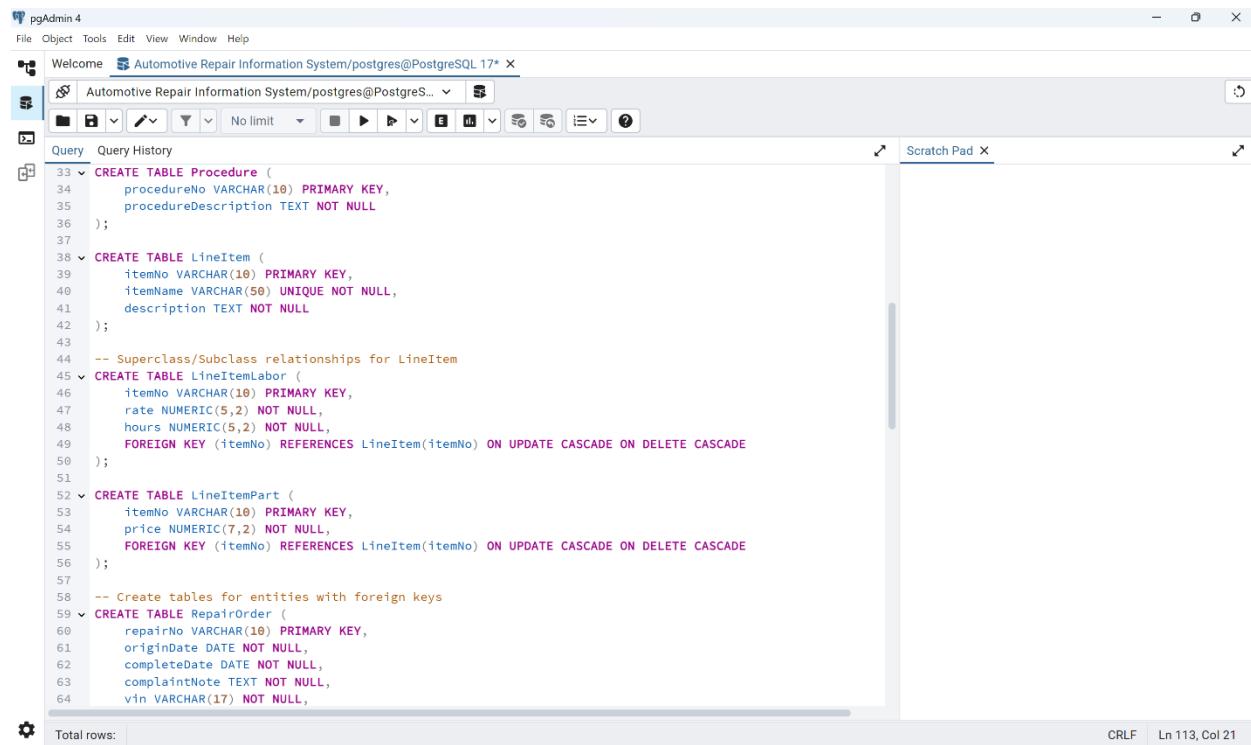
```
CREATE TABLE Procedure (  
    procedureNo VARCHAR(10) PRIMARY KEY,  
    procedureDescription TEXT NOT NULL  
);
```

```
CREATE TABLE LineItem (  
    itemNo VARCHAR(10) PRIMARY KEY,  
    itemName VARCHAR(50) UNIQUE NOT NULL,  
    description TEXT NOT NULL  
);
```

```
-- Superclass/Subclass relationships for LineItem  
CREATE TABLE LineItemLabor (  
    itemNo VARCHAR(10) PRIMARY KEY,  
    rate NUMERIC(5,2) NOT NULL,  
    hours NUMERIC(5,2) NOT NULL,  
    FOREIGN KEY (itemNo) REFERENCES LineItem(itemNo) ON UPDATE CASCADE ON  
DELETE CASCADE
```

```
);
```

```
CREATE TABLE LineItemPart (
    itemNo VARCHAR(10) PRIMARY KEY,
    price NUMERIC(7,2) NOT NULL,
    FOREIGN KEY (itemNo) REFERENCES LineItem(itemNo) ON UPDATE CASCADE ON
DELETE CASCADE
);
```



The screenshot shows the pgAdmin 4 interface with the title bar "pgAdmin 4" and "Welcome Automotive Repair Information System/postgres@PostgreSQL 17*". The main area contains several SQL queries, each starting with a line number and a query identifier (e.g., 33 ~). The queries define various tables and procedures, including LineProcedure, LineItem, LineItemLabor, LineItemPart, and RepairOrder. The "Query" tab is selected, and the "Scratch Pad" tab is visible on the right.

```
33 ~ CREATE TABLE Procedure (
34     procedureNo VARCHAR(10) PRIMARY KEY,
35     procedureDescription TEXT NOT NULL
36 );
37
38 ~ CREATE TABLE LineItem (
39     itemNo VARCHAR(10) PRIMARY KEY,
40     itemName VARCHAR(50) UNIQUE NOT NULL,
41     description TEXT NOT NULL
42 );
43
44 -- Superclass/Subclass relationships for LineItem
45 ~ CREATE TABLE LineItemLabor (
46     itemNo VARCHAR(10) PRIMARY KEY,
47     rate NUMERIC(5,2) NOT NULL,
48     hours NUMERIC(5,2) NOT NULL,
49     FOREIGN KEY (itemNo) REFERENCES LineItem(itemNo) ON UPDATE CASCADE ON DELETE CASCADE
50 );
51
52 ~ CREATE TABLE LineItemPart (
53     itemNo VARCHAR(10) PRIMARY KEY,
54     price NUMERIC(7,2) NOT NULL,
55     FOREIGN KEY (itemNo) REFERENCES LineItem(itemNo) ON UPDATE CASCADE ON DELETE CASCADE
56 );
57
58 -- Create tables for entities with foreign keys
59 ~ CREATE TABLE RepairOrder (
60     repairNo VARCHAR(10) PRIMARY KEY,
61     originDate DATE NOT NULL,
62     completeDate DATE NOT NULL,
63     complaintNote TEXT NOT NULL,
64     vin VARCHAR(17) NOT NULL,
```

-- Create tables for entities with foreign keys

```
CREATE TABLE RepairOrder (
    repairNo VARCHAR(10) PRIMARY KEY,
    originDate DATE NOT NULL,
    completeDate DATE NOT NULL,
    complaintNote TEXT NOT NULL,
    vin VARCHAR(17) NOT NULL,
    staffNo VARCHAR(10) NOT NULL,
    FOREIGN KEY (vin) REFERENCES Vehicle(vin) ON UPDATE CASCADE ON DELETE
CASCADE,
```

```
    FOREIGN KEY (staffNo) REFERENCES Mechanic(staffNo) ON UPDATE CASCADE ON
    DELETE NO ACTION
);

CREATE TABLE Invoice (
    invoiceNo VARCHAR(10) PRIMARY KEY,
    odometerStart INTEGER NOT NULL,
    invoicePrintDate DATE NOT NULL,
    repairNo VARCHAR(10) NOT NULL,
    FOREIGN KEY (repairNo) REFERENCES RepairOrder(repairNo) ON UPDATE CASCADE ON
    DELETE CASCADE
);

-- Subclasses of Invoice
CREATE TABLE InvoiceCost (
    invoiceNo VARCHAR(10) PRIMARY KEY,
    odometerFinish INTEGER NOT NULL,
    paymentDate DATE,
    FOREIGN KEY (invoiceNo) REFERENCES Invoice(invoiceNo) ON UPDATE CASCADE ON
    DELETE CASCADE
);

CREATE TABLE InvoiceEstimate (
    invoiceNo VARCHAR(10) PRIMARY KEY,
    FOREIGN KEY (invoiceNo) REFERENCES Invoice(invoiceNo) ON UPDATE CASCADE ON
    DELETE CASCADE
);
```

The screenshot shows the pgAdmin 4 interface with a query editor window. The title bar reads "pgAdmin 4" and "Welcome Automotive Repair Information System/postgres@PostgreSQL 17*". The main area contains the following PostgreSQL code:

```
-- Create tables for entities with foreign keys
CREATE TABLE RepairOrder (
    repairNo VARCHAR(10) PRIMARY KEY,
    originDate DATE NOT NULL,
    completeDate DATE NOT NULL,
    complaintNote TEXT NOT NULL,
    vin VARCHAR(17) NOT NULL,
    staffNo VARCHAR(10) NOT NULL,
    FOREIGN KEY (vin) REFERENCES Vehicle(vin) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (staffNo) REFERENCES Mechanic(staffNo) ON UPDATE CASCADE ON DELETE NO ACTION
);

CREATE TABLE Invoice (
    invoiceNo VARCHAR(10) PRIMARY KEY,
    odometerStart INTEGER NOT NULL,
    invoicePrintDate DATE NOT NULL,
    repairNo VARCHAR(10) NOT NULL,
    FOREIGN KEY (repairNo) REFERENCES RepairOrder(repairNo) ON UPDATE CASCADE ON DELETE CASCADE
);

-- Subclasses of Invoice
CREATE TABLE InvoiceCost (
    invoiceNo VARCHAR(10) PRIMARY KEY,
    odometerFinish INTEGER NOT NULL,
    paymentDate DATE,
    FOREIGN KEY (invoiceNo) REFERENCES Invoice(invoiceNo) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE InvoiceEstimate (
    invoiceNo VARCHAR(10) PRIMARY KEY,
    FOREIGN KEY (invoiceNo) REFERENCES Invoice(invoiceNo) ON UPDATE CASCADE ON DELETE CASCADE
);
```

-- Create tables for many-to-many relationships

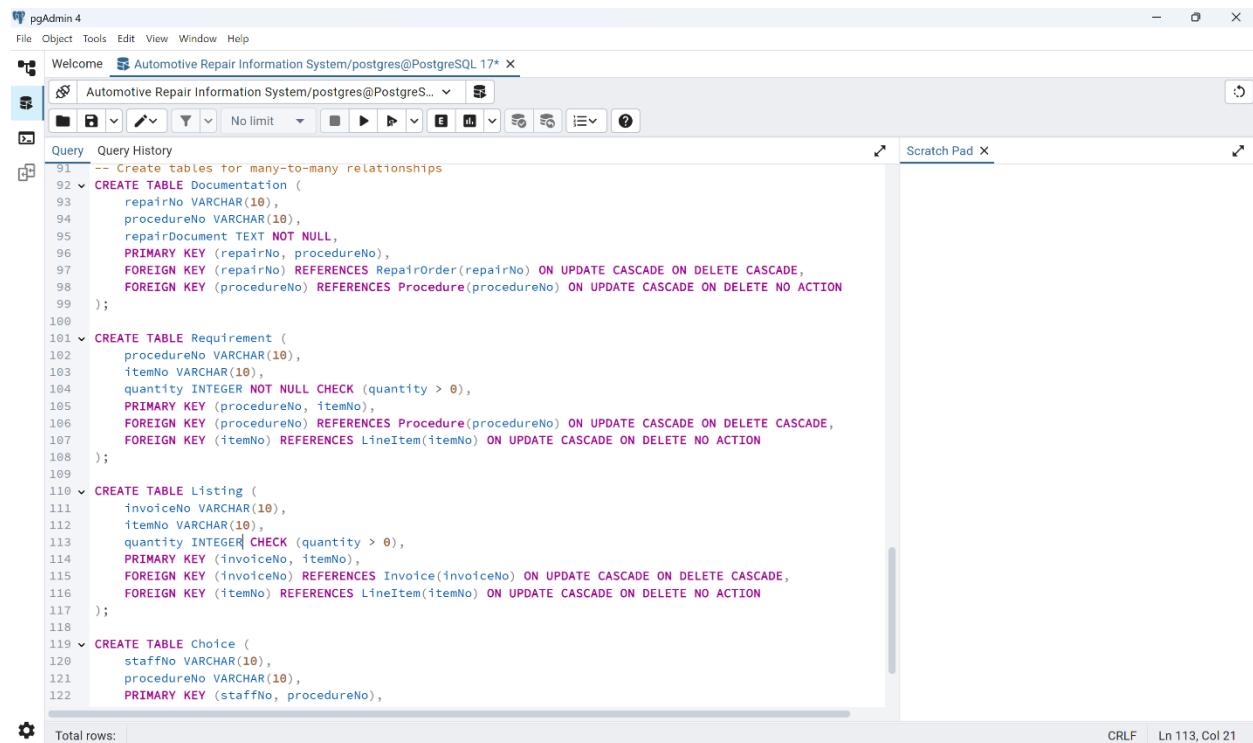
```
CREATE TABLE Documentation (
    repairNo VARCHAR(10),
    procedureNo VARCHAR(10),
    repairDocument TEXT NOT NULL,
    PRIMARY KEY (repairNo, procedureNo),
    FOREIGN KEY (repairNo) REFERENCES RepairOrder(repairNo) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (procedureNo) REFERENCES Procedure(procedureNo) ON UPDATE CASCADE ON DELETE NO ACTION
);
```

CREATE TABLE Requirement (

```
procedureNo VARCHAR(10),
itemNo VARCHAR(10),
quantity INTEGER NOT NULL CHECK (quantity > 0),
PRIMARY KEY (procedureNo, itemNo),
FOREIGN KEY (procedureNo) REFERENCES Procedure(procedureNo) ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (itemNo) REFERENCES LineItem(itemNo) ON UPDATE CASCADE ON DELETE NO ACTION
```

);

```
CREATE TABLE Listing (
    invoiceNo VARCHAR(10),
    itemNo VARCHAR(10),
    quantity INTEGER CHECK (quantity > 0),
    PRIMARY KEY (invoiceNo, itemNo),
    FOREIGN KEY (invoiceNo) REFERENCES Invoice(invoiceNo) ON UPDATE CASCADE ON
    DELETE CASCADE,
    FOREIGN KEY (itemNo) REFERENCES LineItem(itemNo) ON UPDATE CASCADE ON
    DELETE NO ACTION
);
```



The screenshot shows the pgAdmin 4 interface with the 'Query History' tab selected. The query window displays several CREATE TABLE statements for an 'Automotive Repair Information System'. The statements define tables for Documentation, Requirement, Listing, and Choice, each with specific columns, data types, and constraints like primary keys and foreign keys.

```
-- Create tables for many-to-many relationships
CREATE TABLE Documentation (
    repairNo VARCHAR(10),
    procedureNo VARCHAR(10),
    repairDocument TEXT NOT NULL,
    PRIMARY KEY (repairNo, procedureNo),
    FOREIGN KEY (repairNo) REFERENCES RepairOrder(repairNo) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (procedureNo) REFERENCES Procedure(procedureNo) ON UPDATE CASCADE ON DELETE NO ACTION
);

CREATE TABLE Requirement (
    procedureNo VARCHAR(10),
    itemNo VARCHAR(10),
    quantity INTEGER NOT NULL CHECK (quantity > 0),
    PRIMARY KEY (procedureNo, itemNo),
    FOREIGN KEY (procedureNo) REFERENCES Procedure(procedureNo) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (itemNo) REFERENCES LineItem(itemNo) ON UPDATE CASCADE ON DELETE NO ACTION
);

CREATE TABLE Listing (
    invoiceNo VARCHAR(10),
    itemNo VARCHAR(10),
    quantity INTEGER CHECK (quantity > 0),
    PRIMARY KEY (invoiceNo, itemNo),
    FOREIGN KEY (invoiceNo) REFERENCES Invoice(invoiceNo) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (itemNo) REFERENCES LineItem(itemNo) ON UPDATE CASCADE ON DELETE NO ACTION
);

CREATE TABLE Choice (
    staffNo VARCHAR(10),
    procedureNo VARCHAR(10),
    PRIMARY KEY (staffNo, procedureNo),
```

```

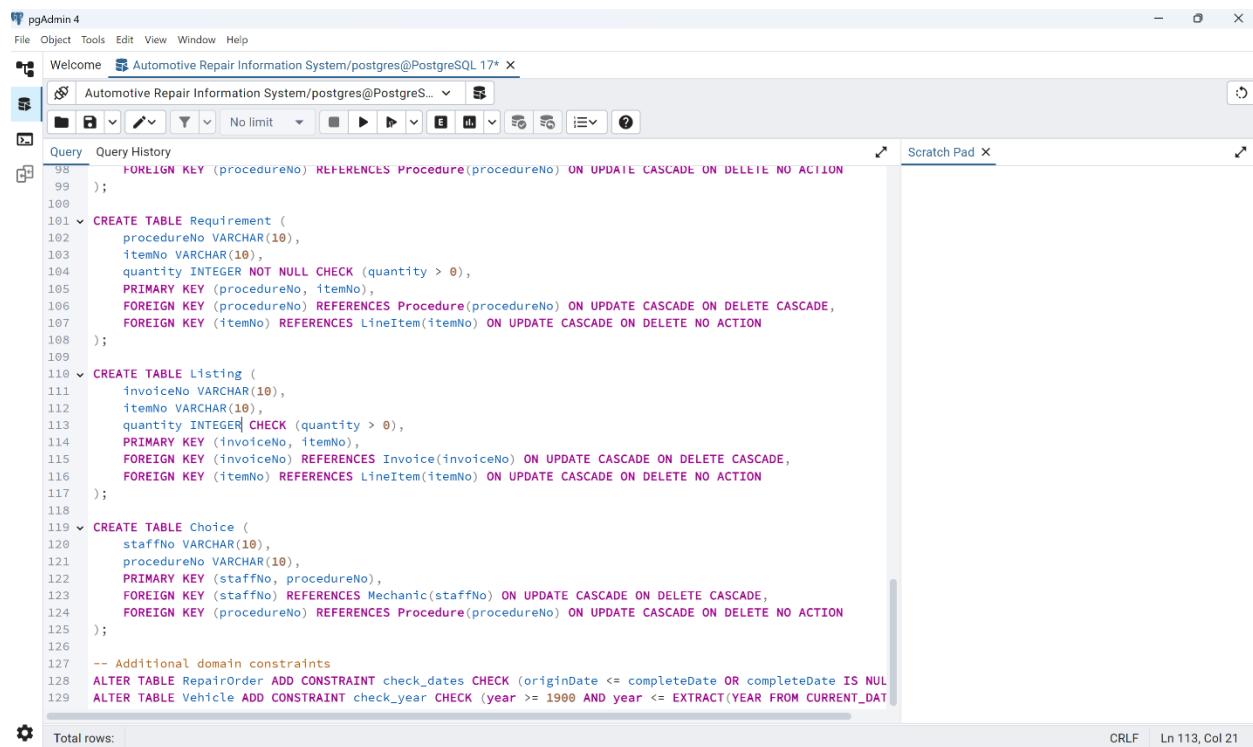
CREATE TABLE Choice (
    staffNo VARCHAR(10),
    procedureNo VARCHAR(10),
    PRIMARY KEY (staffNo, procedureNo),
    FOREIGN KEY (staffNo) REFERENCES Mechanic(staffNo) ON UPDATE CASCADE ON
    DELETE CASCADE,
    FOREIGN KEY (procedureNo) REFERENCES Procedure(procedureNo) ON UPDATE
    CASCADE ON DELETE NO ACTION
);

```

```

-- Additional domain constraints
ALTER TABLE RepairOrder ADD CONSTRAINT check_dates CHECK (originDate <=
completeDate OR completeDate IS NULL);
ALTER TABLE Vehicle ADD CONSTRAINT check_year CHECK (year >= 1900 AND year <=
EXTRACT(YEAR FROM CURRENT_DATE));

```



The screenshot shows the pgAdmin 4 interface with the following details:

- Title Bar:** pgAdmin 4
- Toolbar:** File, Object, Tools, Edit, View, Window, Help
- Query Editor:**
 - Welcome to Automotive Repair Information System/postgres@PostgreSQL 17*
 - Query History tab is selected.
 - Scratch Pad tab is also visible.
 - Code content:

```

98     FOREIGN KEY (procedureNo) REFERENCES Procedure(procedureNo) ON UPDATE CASCADE ON DELETE NO ACTION
99   );
100
101 ✓ CREATE TABLE Requirement (
102     procedureNo VARCHAR(10),
103     itemNo VARCHAR(10),
104     quantity INTEGER NOT NULL CHECK (quantity > 0),
105     PRIMARY KEY (procedureNo, itemNo),
106     FOREIGN KEY (procedureNo) REFERENCES Procedure(procedureNo) ON UPDATE CASCADE ON DELETE CASCADE,
107     FOREIGN KEY (itemNo) REFERENCES LineItem(itemNo) ON UPDATE CASCADE ON DELETE NO ACTION
108   );
109
110 ✓ CREATE TABLE Listing (
111     invoiceNo VARCHAR(10),
112     itemNo VARCHAR(10),
113     quantity INTEGER CHECK (quantity > 0),
114     PRIMARY KEY (invoiceNo, itemNo),
115     FOREIGN KEY (invoiceNo) REFERENCES Invoice(invoiceNo) ON UPDATE CASCADE ON DELETE CASCADE,
116     FOREIGN KEY (itemNo) REFERENCES LineItem(itemNo) ON UPDATE CASCADE ON DELETE NO ACTION
117   );
118
119 ✓ CREATE TABLE Choice (
120     staffNo VARCHAR(10),
121     procedureNo VARCHAR(10),
122     PRIMARY KEY (staffNo, procedureNo),
123     FOREIGN KEY (staffNo) REFERENCES Mechanic(staffNo) ON UPDATE CASCADE ON DELETE CASCADE,
124     FOREIGN KEY (procedureNo) REFERENCES Procedure(procedureNo) ON UPDATE CASCADE ON DELETE NO ACTION
125   );
126
127 -- Additional domain constraints
128 ALTER TABLE RepairOrder ADD CONSTRAINT check_dates CHECK (originDate <= completeDate OR completeDate IS NULL)
129 ALTER TABLE Vehicle ADD CONSTRAINT check_year CHECK (year >= 1900 AND year <= EXTRACT(YEAR FROM CURRENT_DATE))

```
- Status Bar:** Total rows: 0, CRLF, Ln 113, Col 21

These are the results:

```

Data Output Messages Notifications
NOTICE: table "choice" does not exist, skipping
NOTICE: table "listing" does not exist, skipping
NOTICE: table "requirement" does not exist, skipping
NOTICE: table "documentation" does not exist, skipping
NOTICE: table "invoicecost" does not exist, skipping
NOTICE: table "invoiceestimate" does not exist, skipping
NOTICE: table "invoice" does not exist, skipping
NOTICE: table "repairorder" does not exist, skipping
NOTICE: table "lineitemlabor" does not exist, skipping
NOTICE: table "lineitempart" does not exist, skipping
NOTICE: table "lineitem" does not exist, skipping
NOTICE: table "procedure" does not exist, skipping
NOTICE: table "vehicle" does not exist, skipping
NOTICE: table "mechanic" does not exist, skipping
ALTER TABLE
Total rows: Query complete 00:00:00.076
28
29   CONSTRAINT choice_pk PRIMARY KEY (staffno, procedureno),
30   CONSTRAINT choice_procedureno_fkey FOREIGN KEY (procedureno)
31     REFERENCES public.procedure (procedureno) MATCH SIMPLE
32   ON UPDATE CASCADE
33   ON DELETE NO ACTION,
34   CONSTRAINT choice_staffno_fkey FOREIGN KEY (staffno)
35     REFERENCES public.mechanic (staffno) MATCH SIMPLE
36   ON UPDATE CASCADE
37   ON DELETE CASCADE
38 )
39
40 TABLESPACE pg_default;
41
42 ALTER TABLE IF EXISTS public.choice
43   OWNER to postgres;

Data Output Messages Notifications
NOTICE: table "requirement" does not exist, skipping
NOTICE: table "documentation" does not exist, skipping
NOTICE: table "invoicecost" does not exist, skipping
NOTICE: table "invoiceestimate" does not exist, skipping
NOTICE: table "invoice" does not exist, skipping
NOTICE: table "repairorder" does not exist, skipping
NOTICE: table "lineitemlabor" does not exist, skipping
NOTICE: table "lineitempart" does not exist, skipping
NOTICE: table "lineitem" does not exist, skipping
NOTICE: table "procedure" does not exist, skipping
NOTICE: table "vehicle" does not exist, skipping
NOTICE: table "mechanic" does not exist, skipping
ALTER TABLE
Query returned successfully in 76 msec.
Total rows: Query complete 00:00:00.076
CRLF Ln 32, Col 1

```

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- Connections
- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Operators
- Procedures
- Sequences
- Tables (14)
- choice
- documentation
- invoice
- invoicecost
- invoiceestimate
- lineitem
- lineitemlabor
- lineitempart
- listing
- mechanic
- procedure
- repairorder
- requirement
- vehicle

Dashboard Properties SQL Statistics Dependencies Dependents Processes

```

-- Table: public.choice
1
2
3 -- DROP TABLE IF EXISTS public.choice;
4
5 CREATE TABLE IF NOT EXISTS public.choice
6 (
7   staffno character varying(10) COLLATE pg_catalog."default" NOT NULL,
8   procedureno character varying(10) COLLATE pg_catalog."default" NOT NULL,
9   CONSTRAINT choice_pk PRIMARY KEY (staffno, procedureno),
10  CONSTRAINT choice_procedureno_fkey FOREIGN KEY (procedureno)
11    REFERENCES public.procedure (procedureno) MATCH SIMPLE
12   ON UPDATE CASCADE
13   ON DELETE NO ACTION,
14   CONSTRAINT choice_staffno_fkey FOREIGN KEY (staffno)
15     REFERENCES public.mechanic (staffno) MATCH SIMPLE
16   ON UPDATE CASCADE
17   ON DELETE CASCADE
18 )
19
20 TABLESPACE pg_default;
21
22 ALTER TABLE IF EXISTS public.choice
23   OWNER to postgres;

```

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- > B Conditions
- > Domains
- > FTS Configurations
- > FTS Dictionaries
- > Aa FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > 1.3 Sequences
- > Tables (14)
 - > choice
 - > documentation
 - > invoice
 - > invoicecost
 - > invoiceestimate
 - > lineitem
 - > lineitemlabor
 - > lineitempart
 - > listing
 - > mechanic
 - > procedure
 - > repairorder
 - > requirement
 - > vehicle
- > Trigger Functions
- > Types
- > ...

Dashboard X Properties X SQL X Statistics X Dependencies X Dependents X Processes X

```
1 -- Table: public.documentation
2
3 -- DROP TABLE IF EXISTS public.documentation;
4
5 v CREATE TABLE IF NOT EXISTS public.documentation
6 (
7     repairno character varying(10) COLLATE pg_catalog."default" NOT NULL,
8     procedureno character varying(10) COLLATE pg_catalog."default" NOT NULL,
9     repairdocument text COLLATE pg_catalog."default" NOT NULL,
10    CONSTRAINT documentation_pkey PRIMARY KEY (repairno, procedureno),
11    CONSTRAINT documentation_procedureno_fkey FOREIGN KEY (procedureno)
12        REFERENCES public.procedure (procedureno) MATCH SIMPLE
13        ON UPDATE CASCADE
14        ON DELETE NO ACTION,
15    CONSTRAINT documentation_repairno_fkey FOREIGN KEY (repairno)
16        REFERENCES public.repairorder (repairno) MATCH SIMPLE
17        ON UPDATE CASCADE
18        ON DELETE CASCADE
19 )
20
21 TABLESPACE pg_default;
22
23 v ALTER TABLE IF EXISTS public.documentation
24     OWNER to postgres;
```

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- > B Conditions
- > Domains
- > FTS Configurations
- > FTS Dictionaries
- > Aa FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > 1.3 Sequences
- > Tables (14)
 - > choice
 - > documentation
 - > invoice
 - > invoicecost
 - > invoiceestimate
 - > lineitem
 - > lineitemlabor
 - > lineitempart
 - > listing
 - > mechanic
 - > procedure
 - > repairorder
 - > requirement
 - > vehicle
- > Trigger Functions
- > Types
- > ...

Dashboard X Properties X SQL X Statistics X Dependencies X Dependents X Processes X

```
1 -- Table: public.invoice
2
3 -- DROP TABLE IF EXISTS public.invoice;
4
5 v CREATE TABLE IF NOT EXISTS public.invoice
6 (
7     invoiceceno character varying(10) COLLATE pg_catalog."default" NOT NULL,
8     odometerstart integer NOT NULL,
9     invoiceprintdate date NOT NULL,
10    repairno character varying(10) COLLATE pg_catalog."default" NOT NULL,
11    CONSTRAINT invoice_pkey PRIMARY KEY (invoiceceno),
12    CONSTRAINT invoice_repairno_fkey FOREIGN KEY (repairno)
13        REFERENCES public.repairorder (repairno) MATCH SIMPLE
14        ON UPDATE CASCADE
15        ON DELETE CASCADE
16 )
17
18 TABLESPACE pg_default;
19
20 v ALTER TABLE IF EXISTS public.invoice
21     OWNER to postgres;
```

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- > **Commons**
- > Domains
- > FTS Configurations
- > FTS Dictionaries
- > Aa FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- > Tables (14)
 - > choice
 - > documentation
 - > invoice
 - > **invoicecost**
 - > invoiceestimate
 - > lineitem
 - > lineitemlabor
 - > lineitempart
 - > listing
 - > mechanic
 - > procedure
 - > repairorder
 - > requirement
 - > vehicle
- > Trigger Functions
- > Types
- > ...

Dashboard X Properties X **SQL X** Statistics X Dependencies X Dependents X Processes X

```
1 -- Table: public.invoicecost
2
3 -- DROP TABLE IF EXISTS public.invoicecost;
4
5 v CREATE TABLE IF NOT EXISTS public.invoicecost
6 (
7     invoiceno character varying(10) COLLATE pg_catalog."default" NOT NULL,
8     odometerfinish integer NOT NULL,
9     paymentdate date,
10    CONSTRAINT invoicestock_pkey PRIMARY KEY (invoiceno),
11    CONSTRAINT invoicestock_invoiceno_fkey FOREIGN KEY (invoiceno)
12        REFERENCES public.invoice (invoiceno) MATCH SIMPLE
13        ON UPDATE CASCADE
14        ON DELETE CASCADE
15 )
16
17 TABLESPACE pg_default;
18
19 v ALTER TABLE IF EXISTS public.invoicecost
20     OWNER to postgres;
```

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- > **Commons**
- > Domains
- > FTS Configurations
- > FTS Dictionaries
- > Aa FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- > Tables (14)
 - > choice
 - > documentation
 - > invoice
 - > **invoicecost**
 - > **invoiceestimate**
 - > lineitem
 - > lineitemlabor
 - > lineitempart
 - > listing
 - > mechanic
 - > procedure
 - > repairorder
 - > requirement
 - > vehicle
- > Trigger Functions
- > Types
- > ...

Dashboard X Properties X **SQL X** Statistics X Dependencies X Dependents X Processes X

```
1 -- Table: public.invoiceestimate
2
3 -- DROP TABLE IF EXISTS public.invoiceestimate;
4
5 v CREATE TABLE IF NOT EXISTS public.invoiceestimate
6 (
7     invoiceno character varying(10) COLLATE pg_catalog."default" NOT NULL,
8     CONSTRAINT invoiceestimate_pkey PRIMARY KEY (invoiceno),
9     CONSTRAINT invoiceestimate_invoiceno_fkey FOREIGN KEY (invoiceno)
10        REFERENCES public.invoice (invoiceno) MATCH SIMPLE
11        ON UPDATE CASCADE
12        ON DELETE CASCADE
13 )
14
15 TABLESPACE pg_default;
16
17 v ALTER TABLE IF EXISTS public.invoiceestimate
18     OWNER to postgres;
```

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

Dashboard Properties SQL Statistics Dependencies Dependents Processes

```
1 -- Table: public.lineitem
2
3 -- DROP TABLE IF EXISTS public.lineitem;
4
5 ✓ CREATE TABLE IF NOT EXISTS public.lineitem
6 (
7     itemno character varying(10) COLLATE pg_catalog."default" NOT NULL,
8     itemname character varying(50) COLLATE pg_catalog."default" NOT NULL,
9     description text COLLATE pg_catalog."default" NOT NULL,
10    CONSTRAINT lineitem_pkey PRIMARY KEY (itemno),
11    CONSTRAINT lineitem_itemname_key UNIQUE (itemname)
12 )
13
14 TABLESPACE pg_default;
15
16 ✓ ALTER TABLE IF EXISTS public.lineitem
17     OWNER to postgres;
```

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

Dashboard Properties SQL Statistics Dependencies Dependents Processes

```
1 -- Table: public.lineitemlabor
2
3 -- DROP TABLE IF EXISTS public.lineitemlabor;
4
5 ✓ CREATE TABLE IF NOT EXISTS public.lineitemlabor
6 (
7     itemno character varying(10) COLLATE pg_catalog."default" NOT NULL,
8     rate numeric(5,2) NOT NULL,
9     hours numeric(5,2) NOT NULL,
10    CONSTRAINT lineitemlabor_pkey PRIMARY KEY (itemno),
11    CONSTRAINT lineitemlabor_itemno_fkey FOREIGN KEY (itemno)
12        REFERENCES public.lineitem (itemno) MATCH SIMPLE
13        ON UPDATE CASCADE
14        ON DELETE CASCADE
15 )
16
17 TABLESPACE pg_default;
18
19 ✓ ALTER TABLE IF EXISTS public.lineitemlabor
20     OWNER to postgres;
```

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- > Conditions
- > Domains
- > FTS Configurations
- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- > Tables(14)
 - > choice
 - > documentation
 - > invoice
 - > invoicecost
 - > invoiceestimate
 - > lineitem
 - > linitemlabor
 - > lineitempart
 - > listing
 - > mechanic
 - > procedure
 - > repairorder
 - > requirement
 - > vehicle
- > Trigger Functions
- > Types
- > ...

Dashboard X Properties X SQL X Statistics X Dependencies X Dependents X Processes X

```
1 -- Table: public.lineitempart
2
3 -- DROP TABLE IF EXISTS public.lineitempart;
4
5 < CREATE TABLE IF NOT EXISTS public.lineitempart
6 (
7     itemno character varying(10) COLLATE pg_catalog."default" NOT NULL,
8     price numeric(7,2) NOT NULL,
9     CONSTRAINT lineitempart_pkey PRIMARY KEY (itemno),
10    CONSTRAINT lineitempart_itemno_fkey FOREIGN KEY (itemno)
11        REFERENCES public.lineitem (itemno) MATCH SIMPLE
12        ON UPDATE CASCADE
13        ON DELETE CASCADE
14 )
15
16 TABLESPACE pg_default;
17
18 < ALTER TABLE IF EXISTS public.lineitempart
19     OWNER to postgres;
```

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- > Conditions
- > Domains
- > FTS Configurations
- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- > Tables(14)
 - > choice
 - > documentation
 - > invoice
 - > invoicecost
 - > invoiceestimate
 - > lineitem
 - > linitemlabor
 - > lineitempart
 - > listing
 - > mechanic
 - > procedure
 - > repairorder
 - > requirement
 - > vehicle
- > Trigger Functions
- > Types
- > ...

Dashboard X Properties X SQL X Statistics X Dependencies X Dependents X Processes X

```
1 -- Table: public.listing
2
3 -- DROP TABLE IF EXISTS public.listing;
4
5 < CREATE TABLE IF NOT EXISTS public.listing
6 (
7     invoice_no character varying(10) COLLATE pg_catalog."default" NOT NULL,
8     itemno character varying(10) COLLATE pg_catalog."default" NOT NULL,
9     quantity integer,
10    CONSTRAINT listing_pkey PRIMARY KEY (invoice_no, itemno),
11    CONSTRAINT listing_invoice_no_fkey FOREIGN KEY (invoice_no)
12        REFERENCES public.invoice (invoice_no) MATCH SIMPLE
13        ON UPDATE CASCADE
14        ON DELETE CASCADE,
15    CONSTRAINT listing_itemno_fkey FOREIGN KEY (itemno)
16        REFERENCES public.lineitem (itemno) MATCH SIMPLE
17        ON UPDATE CASCADE
18        ON DELETE NO ACTION,
19        CONSTRAINT listing_quantity_check CHECK (quantity > 0)
20 )
21
22 TABLESPACE pg_default;
23
24 < ALTER TABLE IF EXISTS public.listing
25     OWNER to postgres;
```

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- > **Connections**
- > Domains
- > FTS Configurations
- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- > Tables (14)
 - > choice
 - > documentation
 - > invoice
 - > invoicecost
 - > invoiceestimate
 - > lineitem
 - > lineitemlabor
 - > lineitempart
 - > listing
 - > **mechanic**
 - > procedure
 - > repairorder
 - > requirement
 - > vehicle
- > Trigger Functions
- > Types
- > Views

Dashboard X Properties X SQL X Statistics X Dependencies X Dependents X Processes X

```
1 -- Table: public.mechanic
2
3 -- DROP TABLE IF EXISTS public.mechanic;
4
5 v CREATE TABLE IF NOT EXISTS public.mechanic
6 (
7     staffno character varying(10) COLLATE pg_catalog."default" NOT NULL,
8     fname character varying(50) COLLATE pg_catalog."default" NOT NULL,
9     lname character varying(50) COLLATE pg_catalog."default" NOT NULL,
10    CONSTRAINT mechanic_pkey PRIMARY KEY (staffno)
11 )
12
13 TABLESPACE pg_default;
14
15 v ALTER TABLE IF EXISTS public.mechanic
16     OWNER to postgres;
```

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- > **Connections**
- > Domains
- > FTS Configurations
- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- > Tables (14)
 - > choice
 - > documentation
 - > invoice
 - > invoicecost
 - > invoiceestimate
 - > lineitem
 - > lineitemlabor
 - > lineitempart
 - > listing
 - > mechanic
 - > **procedure**
 - > repairorder
 - > requirement
 - > vehicle
- > Trigger Functions
- > Types
- > Views

Dashboard X Properties X SQL X Statistics X Dependencies X Dependents X Processes X

```
1 -- Table: public.procedure
2
3 -- DROP TABLE IF EXISTS public.procedure;
4
5 v CREATE TABLE IF NOT EXISTS public.procedure
6 (
7     procedureno character varying(10) COLLATE pg_catalog."default" NOT NULL,
8     proceduredescription text COLLATE pg_catalog."default" NOT NULL,
9     CONSTRAINT procedure_pkey PRIMARY KEY (procedureno)
10 )
11
12 TABLESPACE pg_default;
13
14 v ALTER TABLE IF EXISTS public.procedure
15     OWNER to postgres;
```

```
-- Table: public.repairorder
-- DROP TABLE IF EXISTS public.repairorder;

CREATE TABLE IF NOT EXISTS public.repairorder
(
    repairno character varying(10) COLLATE pg_catalog."default" NOT NULL,
    origindate date NOT NULL,
    completedate date NOT NULL,
    complaintnote text COLLATE pg_catalog."default" NOT NULL,
    vin character varying(17) COLLATE pg_catalog."default" NOT NULL,
    staffno character varying(10) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT repairorder_pkey PRIMARY KEY (repairno),
    CONSTRAINT repairorder_staffno_fkey FOREIGN KEY (staffno)
        REFERENCES public.mechanic (staffno) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE NO ACTION,
    CONSTRAINT repairorder_vin_fkey FOREIGN KEY (vin)
        REFERENCES public.vehicle (vin) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT check_dates CHECK (origindate <= completedate OR completedate IS NULL)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.repairorder
    OWNER to postgres;
```

```
-- Table: public.requirement
-- DROP TABLE IF EXISTS public.requirement;

CREATE TABLE IF NOT EXISTS public.requirement
(
    procedureno character varying(10) COLLATE pg_catalog."default" NOT NULL,
    itemno character varying(10) COLLATE pg_catalog."default" NOT NULL,
    quantity integer NOT NULL,
    CONSTRAINT requirement_pkey PRIMARY KEY (procedureno, itemno),
    CONSTRAINT requirement_itemno_fkey FOREIGN KEY (itemno)
        REFERENCES public.lineitem (itemno) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE NO ACTION,
    CONSTRAINT requirement_procedureno_fkey FOREIGN KEY (procedureno)
        REFERENCES public.procedure (procedureno) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT requirement_quantity_check CHECK (quantity > 0)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.requirement
    OWNER to postgres;
```

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer Dashboard Properties SQL Statistics Dependencies Dependents Processes

Correlations Domains FTS Configurations FTS Dictionaries FTS Parsers FTS Templates Foreign Tables Functions Materialized Views Operators Procedures Sequences Tables(14) choice documentation invoice invoicecost invoiceestimate linelitem linelitemlabor linelitempart listing mechanic procedure repairorder requirement vehicle Trigger Functions Types

```
1 -- Table: public.vehicle
2
3 -- DROP TABLE IF EXISTS public.vehicle;
4
5 CREATE TABLE IF NOT EXISTS public.vehicle
6 (
7     vin character varying(17) COLLATE pg_catalog."default" NOT NULL,
8     make character varying(50) COLLATE pg_catalog."default" NOT NULL,
9     year integer NOT NULL,
10    model character varying(50) COLLATE pg_catalog."default" NOT NULL,
11    CONSTRAINT vehicle_pkey PRIMARY KEY (vin),
12    CONSTRAINT check_year CHECK (year >= 1900 AND year::numeric <= EXTRACT(year FROM CURRENT_DATE))
13 )
14
15 TABLESPACE pg_default;
16
17 ALTER TABLE IF EXISTS public.vehicle
18     OWNER to postgres;
```

Test Data

-- Sample data for Auto Repair Shop Management System

-- Insert sample vehicles

```
INSERT INTO Vehicle (vin, make, year, model) VALUES
('1HGCM82633A123456', 'Honda', 2018, 'Accord'),
('5FNRL38225B123456', 'Toyota', 2020, 'Camry'),
('2HGES165X9H123456', 'Ford', 2019, 'F-150'),
('1FAHP2EW7CG123456', 'Chevrolet', 2021, 'Malibu'),
('WBADT63433C123456', 'BMW', 2017, '328i');
```

-- Insert sample mechanics

```
INSERT INTO Mechanic (staffNo, fName, lName) VALUES
('MECH001', 'David', 'Jones'),
('MECH002', 'Lisa', 'Thompson'),
('MECH003', 'Carlos', 'Rodriguez'),
('MECH004', 'Jessica', 'Miller'),
('MECH005', 'Kevin', 'Anderson');
```

-- Insert sample procedures

```
INSERT INTO Procedure (procedureNo, procedureDescription) VALUES
('PROC001', 'Oil Change'),
('PROC002', 'Brake Pad Replacement'),
('PROC003', 'Tire Rotation'),
('PROC004', 'Engine Tune-Up'),
('PROC005', 'Transmission Fluid Change');
```

The screenshot shows the pgAdmin 4 interface with a query editor containing SQL code. The code is organized into several sections with comments starting with '--'. It includes sample data for vehicles, mechanics, procedures, and line items. The pgAdmin interface has a toolbar at the top with various icons for database management, and the main window shows the query text and its execution status.

```
1 -- Sample data for Auto Repair Shop Management System
2
3 -- Insert sample vehicles
4 INSERT INTO Vehicle (vin, make, year, model) VALUES
5 ('1HGCM82633A123456', 'Honda', 2018, 'Accord'),
6 ('5FNR138225B123456', 'Toyota', 2020, 'Camry'),
7 ('2HGES165X9H123456', 'Ford', 2019, 'F-150'),
8 ('1FAHP2E7CG123456', 'Chevrolet', 2021, 'Malibu'),
9 ('WBADT63433C123456', 'BMW', 2017, '328i');
10
11 -- Insert sample mechanics
12 INSERT INTO Mechanic (staffNo, fName, lName) VALUES
13 ('MECH001', 'David', 'Jones'),
14 ('MECH002', 'Lisa', 'Thompson'),
15 ('MECH003', 'Carlos', 'Rodriguez'),
16 ('MECH004', 'Jessica', 'Miller'),
17 ('MECH005', 'Kevin', 'Anderson');
18
19 -- Insert sample procedures
20 INSERT INTO Procedure (procedureNo, procedureDescription) VALUES
21 ('PROC001', 'Oil Change'),
22 ('PROC002', 'Brake Pad Replacement'),
23 ('PROC003', 'Tire Rotation'),
24 ('PROC004', 'Engine Tune-Up'),
25 ('PROC005', 'Transmission Fluid Change');
26
27 -- Insert sample line items
```

-- Insert sample line items

INSERT INTO LineItem (itemNo, itemName, description) VALUES

-- Parts

('PART001', 'Oil Filter', 'Standard oil filter for most vehicles'),
('PART002', 'Engine Oil (5W-30)', 'Synthetic oil 5W-30 grade, 1 quart'),
('PART003', 'Brake Pads (Front)', 'Standard front brake pads set'),
('PART004', 'Brake Pads (Rear)', 'Standard rear brake pads set'),
('PART005', 'Air Filter', 'Standard air filter for most vehicles'),

-- Labor

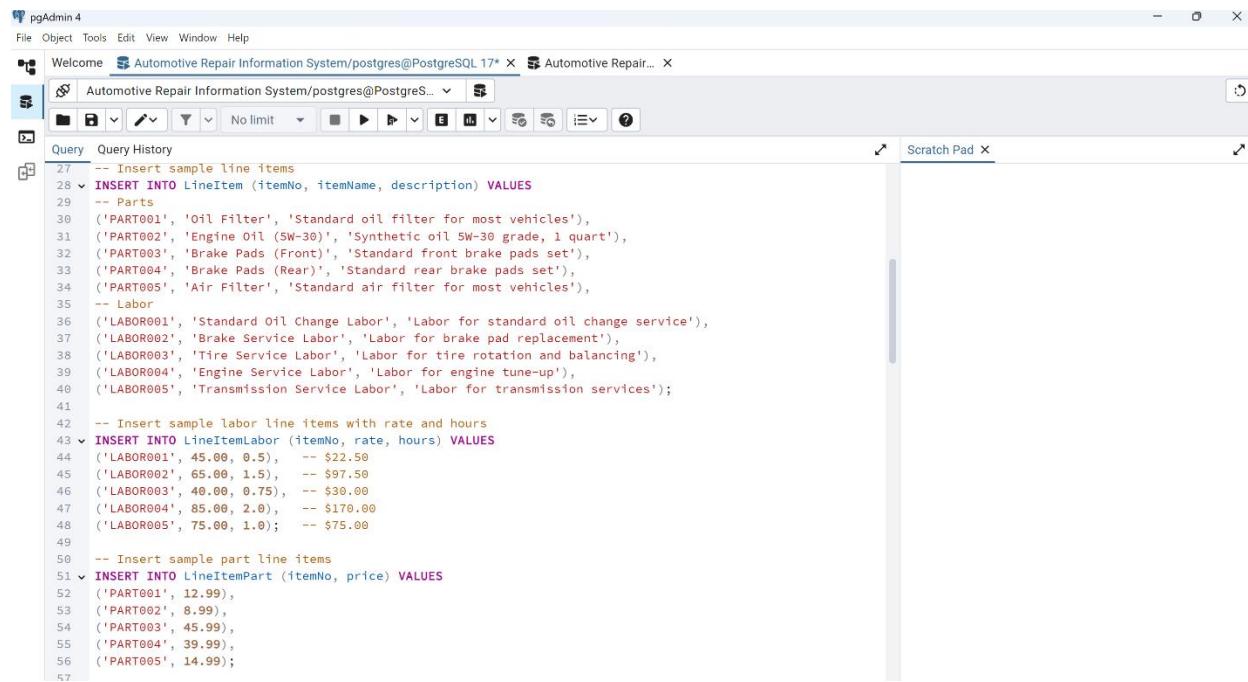
('LABOR001', 'Standard Oil Change Labor', 'Labor for standard oil change service'),
('LABOR002', 'Brake Service Labor', 'Labor for brake pad replacement'),
('LABOR003', 'Tire Service Labor', 'Labor for tire rotation and balancing'),
('LABOR004', 'Engine Service Labor', 'Labor for engine tune-up'),
('LABOR005', 'Transmission Service Labor', 'Labor for transmission services');

-- Insert sample labor line items with rate and hours

INSERT INTO LineItemLabor (itemNo, rate, hours) VALUES

('LABOR001', 45.00, 0.5), -- \$22.50
('LABOR002', 65.00, 1.5), -- \$97.50
('LABOR003', 40.00, 0.75), -- \$30.00
('LABOR004', 85.00, 2.0), -- \$170.00
('LABOR005', 75.00, 1.0); -- \$75.00

```
-- Insert sample part line items
INSERT INTO LineItemPart (itemNo, price) VALUES
('PART001', 12.99),
('PART002', 8.99),
('PART003', 45.99),
('PART004', 39.99),
('PART005', 14.99);
```



The screenshot shows the pgAdmin 4 interface with the title bar "pgAdmin 4" and "Welcome Automotive Repair Information System/postgres@PostgreSQL 17*". The main window displays the following SQL script:

```

27 -- Insert sample line items
28 v INSERT INTO LineItem (itemNo, itemName, description) VALUES
29 -- Parts
30 ('PART001', 'Oil Filter', 'Standard oil filter for most vehicles'),
31 ('PART002', 'Engine Oil (SW-30)', 'Synthetic oil SW-30 grade, 1 quart'),
32 ('PART003', 'Brake Pads (Front)', 'Standard front brake pads set'),
33 ('PART004', 'Brake Pads (Rear)', 'Standard rear brake pads set'),
34 ('PART005', 'Air Filter', 'Standard air filter for most vehicles'),
35 -- Labor
36 ('LABOR001', 'Standard Oil Change Labor', 'Labor for standard oil change service'),
37 ('LABOR002', 'Brake Service Labor', 'Labor for brake pad replacement'),
38 ('LABOR003', 'Tire Service Labor', 'Labor for tire rotation and balancing'),
39 ('LABOR004', 'Engine Service Labor', 'Labor for engine tune-up'),
40 ('LABOR005', 'Transmission Service Labor', 'Labor for transmission services');
41
42 -- Insert sample labor line items with rate and hours
43 v INSERT INTO LineItemLabor (itemNo, rate, hours) VALUES
44 ('LABOR001', 45.00, 0.5),    -- $22.50
45 ('LABOR002', 65.00, 1.5),   -- $97.50
46 ('LABOR003', 40.00, 0.75),  -- $30.00
47 ('LABOR004', 85.00, 2.0),   -- $170.00
48 ('LABOR005', 75.00, 1.0);  -- $75.00
49
50 -- Insert sample part line items
51 v INSERT INTO LineItemPart (itemNo, price) VALUES
52 ('PART001', 12.99),
53 ('PART002', 8.99),
54 ('PART003', 45.99),
55 ('PART004', 39.99),
56 ('PART005', 14.99);
57

```

-- Insert sample repair orders

```
INSERT INTO RepairOrder (repairNo, originDate, completeDate, complaintNote, vin, staffNo) VALUES
('REP001', '2024-05-01', '2024-05-01', 'Routine oil change and inspection',
'1HGCM82633A123456', 'MECH001'),
('REP002', '2024-05-05', '2024-05-06', 'Squeaking brakes when stopping',
'5FNRL38225B123456', 'MECH002'),
('REP003', '2024-05-10', '2024-05-10', 'Tire wear uneven', '2HGES165X9H123456',
'MECH003'),
('REP004', '2024-06-15', '2024-06-17', 'Engine running rough', '1FAHP2EW7CG123456',
'MECH001'),
```

```
('REP005', '2024-07-20', '2024-07-20', 'Transmission shifting hard', 'WBADT63433C123456',
'MECH005'),
('REP006', '2024-08-01', '2024-08-02', 'Check engine light on', '1HGCM82633A123456',
'MECH004');
```

-- Insert sample invoices

```
INSERT INTO Invoice (invoiceNo, odometerStart, invoicePrintDate, repairNo) VALUES
('INV001', 35000, '2024-05-01', 'REP001'),
('INV002', 42500, '2024-05-06', 'REP002'),
('INV003', 28750, '2024-05-10', 'REP003'),
('INV004', 51200, '2024-06-17', 'REP004'),
('INV005', 67300, '2024-07-20', 'REP005'),
('INV006', 37500, '2024-08-01', 'REP006');
```

-- Insert sample invoice costs (completed invoices)

```
INSERT INTO InvoiceCost (invoiceNo, odometerFinish, paymentDate) VALUES
('INV001', 35010, '2024-05-01'),
('INV002', 42510, '2024-05-06'),
('INV003', 28760, '2024-05-10'),
('INV004', 51220, '2024-06-17'),
('INV005', 67310, '2024-07-20');
```

-- Insert sample invoice estimates

```
INSERT INTO InvoiceEstimate (invoiceNo) VALUES
('INV006');
```

```

File Object Tools Edit View Window Help
Welcome to Automotive Repair Information System/postgres@PostgreSQL 17 X Automotive Repair...
Query Query History Scratch Pad X
58 -- Insert sample repair orders
59 ✓ INSERT INTO RepairOrder (repairNo, originDate, completeDate, complaintNote, vin, staffNo) VALUES
60 ('REP001', '2024-05-01', '2024-05-01', 'Routine oil change and inspection', '1HGCM82633A123456', 'MECH001')
61 ('REP002', '2024-05-05', '2024-05-06', 'Squeaking brakes when stopping', '5FNR138225B123456', 'MECH002'),
62 ('REP003', '2024-05-10', '2024-05-10', 'Tire wear uneven', '2HGES165X9H123456', 'MECH003'),
63 ('REP004', '2024-06-15', '2024-06-17', 'Engine running rough', '1FAHP2EWT7CG123456', 'MECH001'),
64 ('REP005', '2024-07-20', '2024-07-20', 'Transmission shifting hard', 'WBADT6343C123456', 'MECH005'),
65 ('REP006', '2024-08-01', '2024-08-02', 'Check engine light on', '1HGCM82633A123456', 'MECH004');
66
67 -- Insert sample invoices
68 ✓ INSERT INTO Invoice (invoiceNo, odometerStart, invoicePrintDate, repairNo) VALUES
69 ('INV001', 35000, '2024-05-01', 'REP001'),
70 ('INV002', 42500, '2024-05-06', 'REP002'),
71 ('INV003', 28750, '2024-05-10', 'REP003'),
72 ('INV004', 51200, '2024-06-17', 'REP004'),
73 ('INV005', 67300, '2024-07-20', 'REP005'),
74 ('INV006', 37500, '2024-08-01', 'REP006');
75
76 -- Insert sample invoice costs (completed invoices)
77 ✓ INSERT INTO InvoiceCosts (invoiceNo, odometerFinish, paymentDate) VALUES
78 ('INV001', 35010, '2024-05-01'),
79 ('INV002', 42510, '2024-05-06'),
80 ('INV003', 28760, '2024-05-10'),
81 ('INV004', 51220, '2024-06-17'),
82 ('INV005', 67310, '2024-07-20');
83
84 -- Insert sample invoice estimates
85 ✓ INSERT INTO InvoiceEstimate (invoiceNo) VALUES
86 ('INV006');
87
88 -- Insert procedure requirements (which parts/labor each procedure needs)

```

-- Insert procedure requirements (which parts/labor each procedure needs)

INSERT INTO Requirement (procedureNo, itemNo, quantity) VALUES

-- Oil Change requirements

('PROC001', 'PART001', 1), -- Oil Filter

('PROC001', 'PART002', 5), -- 5 quarts of oil

('PROC001', 'LABOR001', 1), -- Oil change labor

-- Brake Pad Replacement requirements

('PROC002', 'PART003', 1), -- Front brake pads

('PROC002', 'PART004', 1), -- Rear brake pads

('PROC002', 'LABOR002', 1), -- Brake service labor

-- Tire Rotation requirements

('PROC003', 'LABOR003', 1), -- Tire service labor

-- Engine Tune-Up requirements

('PROC004', 'PART005', 1), -- Air filter

('PROC004', 'LABOR004', 1), -- Engine service labor

-- Transmission Fluid Change requirements

('PROC005', 'PART002', 6), -- 6 quarts of oil type fluid

```
('PROC005', 'LABOR005', 1); -- Transmission service labor

-- Insert invoice listings (which parts/labor are on each invoice)
INSERT INTO Listing (invoiceNo, itemNo, quantity) VALUES
-- Invoice 1: Oil Change
('INV001', 'PART001', 1),
('INV001', 'PART002', 5),
('INV001', 'LABOR001', 1),

-- Invoice 2: Brake Pad Replacement
('INV002', 'PART003', 1),
('INV002', 'PART004', 1),
('INV002', 'LABOR002', 1),

-- Invoice 3: Tire Rotation
('INV003', 'LABOR003', 1),

-- Invoice 4: Engine Tune-Up
('INV004', 'PART005', 1),
('INV004', 'LABOR004', 1),

-- Invoice 5: Transmission Fluid Change
('INV005', 'PART002', 6),
('INV005', 'LABOR005', 1),

-- Invoice 6: Diagnostic (Estimate)
('INV006', 'LABOR001', 0.5);

-- Insert sample documentation
INSERT INTO Documentation (repairNo, procedureNo, repairDocument) VALUES
('REP001', 'PROC001', 'Oil change performed using synthetic 5W-30. All fluid levels checked.'),
('REP002', 'PROC002', 'Replaced front and rear brake pads. Inspected rotors which are still in good condition.'),
('REP003', 'PROC003', 'Rotated tires and checked pressure. Recommended alignment for next visit.'),
('REP004', 'PROC004', 'Performed full engine tune-up. Replaced spark plugs and air filter.'),
('REP005', 'PROC005', 'Flushed and replaced transmission fluid. Adjusted shift points.');
```

```
-- Insert procedure requirements (which parts/labor each procedure needs)
INSERT INTO Requirement (procedureNo, itemNo, quantity) VALUES
-- Oil Change requirements
('PROC001', 'PART001', 1), -- Oil Filter
('PROC001', 'PART002', 5), -- 5 quarts of oil
('PROC001', 'LABOR001', 1), -- Oil change labor
-- Brake Pad Replacement requirements
('PROC002', 'PART003', 1), -- Front brake pads
('PROC002', 'PART004', 1), -- Rear brake pads
('PROC002', 'LABOR002', 1), -- Brake service labor
-- Tire Rotation requirements
('PROC003', 'LABOR003', 1), -- Tire service labor
-- Engine Tune-Up requirements
('PROC004', 'PART005', 1), -- Air filter
('PROC004', 'LABOR004', 1), -- Engine service labor
-- Transmission Fluid Change requirements
('PROC005', 'PART002', 6), -- 6 quarts of oil type fluid
('PROC005', 'LABOR005', 1); -- Transmission service labor
-- Insert invoice listings (which parts/labor are on each invoice)
INSERT INTO Listing (invoiceNo, itemNo, quantity)
VALUES
-- Invoice 1: Oil Change
('INV001', 'PART001', 1),
('INV001', 'PART002', 5),
('INV001', 'LABOR001', 1),
-- Invoice 2: Brake Pad Replacement
('INV002', 'PART003', 1),
('INV002', 'PART004', 1),
('INV002', 'LABOR002', 1),
-- Invoice 3: Engine Tune-Up
('INV003', 'PART005', 1),
('INV003', 'LABOR004', 1),
-- Invoice 4: Transmission Fluid Change
('INV004', 'PART002', 6),
('INV004', 'LABOR005', 1);
-- Invoice 5: Tire Rotation
('INV005', 'LABOR003', 1);
```

```
-- Invoice 2: Brake Pad Replacement
('INV002', 'PART003', 1),
('INV002', 'PART004', 1),
('INV002', 'LABOR002', 1),
-- Invoice 3: Tire Rotation
('INV003', 'LABOR003', 1),
-- Invoice 4: Engine Tune-Up
('INV004', 'PART005', 1),
('INV004', 'LABOR004', 1),
-- Invoice 5: Transmission Fluid Change
('INV005', 'PART002', 6),
('INV005', 'LABOR005', 1),
-- Invoice 6: Diagnostic (Estimate)
('INV006', 'LABOR001', 0.5);
-- Insert sample documentation
INSERT INTO Documentation (repairNo, procedureNo, repairDocument) VALUES
('REP001', 'PROC001', 'Oil change performed using synthetic 5W-30. All fluid levels checked.'),
('REP002', 'PROC002', 'Replaced front and rear brake pads. Inspected rotors which are still in good condition.'),
('REP003', 'PROC003', 'Rotated tires and checked pressure. Recommended alignment for next visit.'),
('REP004', 'PROC004', 'Performed full engine tune-up. Replaced spark plugs and air filter.'),
('REP005', 'PROC005', 'Flushed and replaced transmission fluid. Adjusted shift points.');
-- Insert mechanic specializations (which mechanics can do which procedures)
INSERT INTO Choice (staffNo, procedureNo) VALUES
('MECH001', 'PROC001'),
```

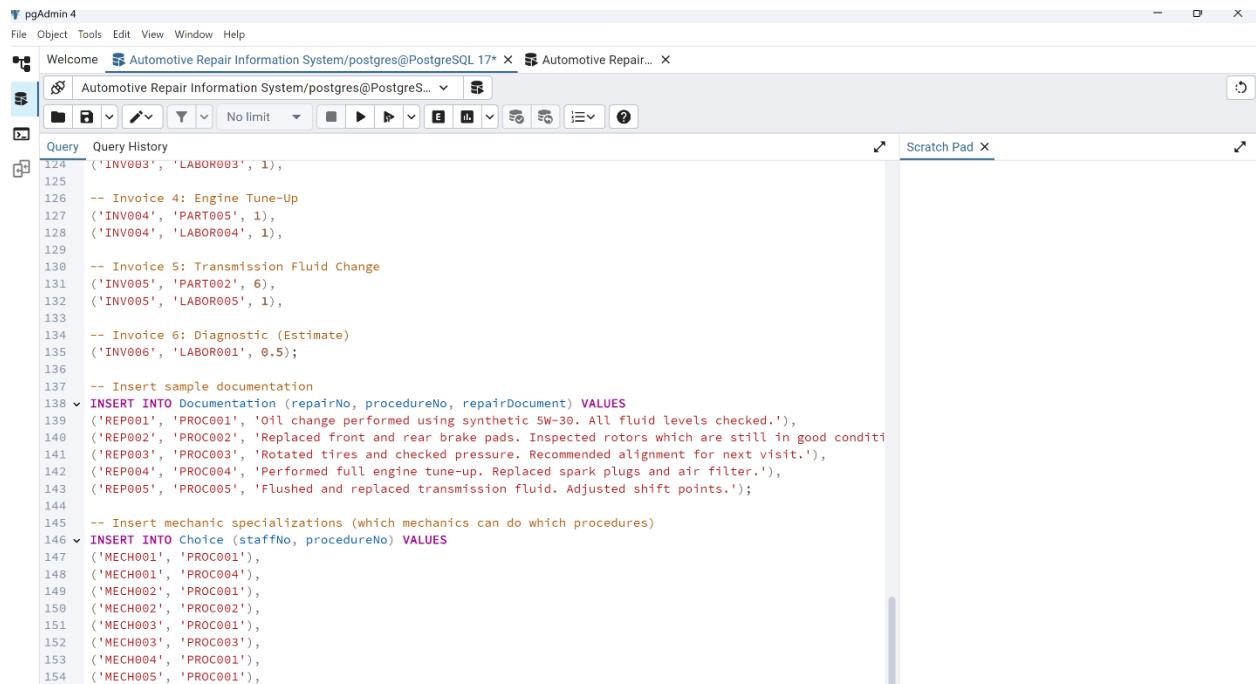
-- Insert mechanic specializations (which mechanics can do which procedures)

INSERT INTO Choice (staffNo, procedureNo) VALUES

('MECH001', 'PROC001'),

('MECH001', 'PROC004'),

```
('MECH002', 'PROC001'),
('MECH002', 'PROC002'),
('MECH003', 'PROC001'),
('MECH003', 'PROC003'),
('MECH004', 'PROC001'),
('MECH005', 'PROC001'),
('MECH005', 'PROC005');
```



The screenshot shows the pgAdmin 4 application window. The title bar reads "pgAdmin 4" and "Welcome - Automotive Repair Information System/postgres@PostgreSQL 17*". The main area is a "Query" editor tab containing the following PostgreSQL code:

```

124 ('INV003', 'LABOR003', 1),
125
126 -- Invoice 4: Engine Tune-Up
127 ('INV004', 'PART005', 1),
128 ('INV004', 'LABOR004', 1),
129
130 -- Invoice 5: Transmission Fluid Change
131 ('INV005', 'PART002', 6),
132 ('INV005', 'LABOR005', 1),
133
134 -- Invoice 6: Diagnostic (Estimate)
135 ('INV006', 'LABOR001', 0.5);
136
137 -- Insert sample documentation
138 ✓ INSERT INTO Documentation (repairNo, procedureNo, repairDocument) VALUES
139 ('REP001', 'PROC001', 'Oil change performed using synthetic 5W-30. All fluid levels checked.'),
140 ('REP002', 'PROC002', 'Replaced front and rear brake pads. Inspected rotors which are still in good condition.'),
141 ('REP003', 'PROC003', 'Rotated tires and checked pressure. Recommended alignment for next visit.'),
142 ('REP004', 'PROC004', 'Performed full engine tune-up. Replaced spark plugs and air filter.'),
143 ('REP005', 'PROC005', 'Flushed and replaced transmission fluid. Adjusted shift points.');
144
145 -- Insert mechanic specializations (which mechanics can do which procedures)
146 ✓ INSERT INTO Choice (staffNo, procedureNo) VALUES
147 ('MECH001', 'PROC001'),
148 ('MECH001', 'PROC004'),
149 ('MECH002', 'PROC001'),
150 ('MECH002', 'PROC002'),
151 ('MECH003', 'PROC001'),
152 ('MECH003', 'PROC003'),
153 ('MECH004', 'PROC001'),
154 ('MECH005', 'PROC001');

```

Resulting Tables:

pgAdmin 4

File Object Tools Edit View Window Help

Welcome Automotive Repair... Automotive Repair... public.choice/Automotive Repair Information System/postgres@PostgreSQL 17

public.choice/Automotive Repair Information System/postgres@PostgreSQL 17

Query Query History Scratch Pad

```
1 ✓ SELECT * FROM public.choice
2 ORDER BY staffno ASC, procedureno ASC
```

Data Output Messages Notifications

staffno	procedureno
1 MECH001	PROC001
2 MECH001	PROC004
3 MECH002	PROC001
4 MECH002	PROC002
5 MECH003	PROC001
6 MECH003	PROC003
7 MECH004	PROC001
8 MECH005	PROC001
9 MECH005	PROC005

Showing rows: 1 to 9 | Page No: 1 of 1 | < << > >> | ↻

pgAdmin 4

File Object Tools Edit View Window Help

Welcome Automotive Repair... Automotive Repair... public.choice/Auto... public.documentation/Automotive Repair Information System/postgres@PostgreSQL 17

public.documentation/Automotive Repair Information System/postgres@PostgreSQL 17

Query Query History Scratch Pad

```
1 ✓ SELECT * FROM public.documentation
2 ORDER BY repairno ASC, procedureno ASC
```

Data Output Messages Notifications

repairno	procedureno	repairdocument
1 REP001	PROC001	Oil change performed using synthetic 5W-30. All fluid levels checked.
2 REP002	PROC002	Replaced front and rear brake pads. Inspected rotors which are still in good condition.
3 REP003	PROC003	Rotated tires and checked pressure. Recommended alignment for next visit.
4 REP004	PROC004	Performed full engine tune-up. Replaced spark plugs and air filter.
5 REP005	PROC005	Flushed and replaced transmission fluid. Adjusted shift points.

Showing rows: 1 to 5 | Page No: 1 of 1 | < << > >> | ↻

✓ Successfully run. Total query runtime: 159 msec. 5 rows affected. ✘

pgAdmin 4

File Object Tools Edit View Window Help

Welcome Automotive Repair... Automotive Repair... public.choice/Auto... public.documentat... public.invoice/Automotive Repair Information System/postgres@PostgreSQL 17

public.invoice/Automotive Repair Information System/postgres@PostgreSQL 17

Query Query History

```
1 ✓ SELECT * FROM public.invoice
2 ORDER BY invoceno ASC
```

Data Output Messages Notifications

	invoceno [PK] character varying (10)	odometerstart integer	invoiceprintdate date	repairo character varying (10)
1	INV001	35000	2024-05-01	REP001
2	INV002	42500	2024-05-06	REP002
3	INV003	28750	2024-05-10	REP003
4	INV004	51200	2024-06-17	REP004
5	INV005	67300	2024-07-20	REP005
6	INV006	37500	2024-08-01	REP006

Showing rows: 1 to 6 Page No: 1 of 1 |<|<|>|>|>>

✓ Successfully run. Total query runtime: 77 msec. 6 rows affected. X

pgAdmin 4

File Object Tools Edit View Window Help

public.invoicecost/Automotive Repair Information System/postgres@PostgreSQL 17

public.invoicecost/Automotive Repair Information System/postgres@PostgreSQL 17

Query Query History

```
1 ✓ SELECT * FROM public.invoicecost
2 ORDER BY invoceno ASC
```

Data Output Messages Notifications

	invoceno [PK] character varying (10)	odometerfinish integer	paymentdate date
1	INV001	35010	2024-05-01
2	INV002	42510	2024-05-06
3	INV003	28760	2024-05-10
4	INV004	51220	2024-06-17
5	INV005	67310	2024-07-20

Showing rows: 1 to 5 Page No: 1

pgAdmin 4

File Object Tools Edit View Window Help

public.choice/Auto... X public.documentat... X public.invoice/Aut... X public.invoicecost... X public.invoiceestimate/Automotive Repair Information System/postgres@PostgreSQL 17 X

public.invoiceestimate/Automotive Repair Information System/...

Query Query History

```
1 v SELECT * FROM public.invoiceestimate
2   ORDER BY invoceno ASC
```

Data Output Messages Notifications

invoceno [Pk] character varying (10)

invoceno
1 INV006

Showing rows: 1 to 1 Page No: 1 of 1

✓ Successfully run. Total query runtime: 68 msec. 1 rows affected. X

pgAdmin 4

File Object Tools Edit View Window Help

/Auto... X public.documentat... X public.invoice/Aut... X public.invoicecost... X public.invoiceesti... X public.lineitem/Automotive Repair Information System/postgres@PostgreSQL 17 X

public.lineitem/Automotive Repair Information System/postgres@PostgreSQL 17 X

Query Query History

```
1 v SELECT * FROM public.lineitem
2   ORDER BY itemno ASC
```

Data Output Messages Notifications

Itemno	Itemname	Description
1 LABOR001	Standard Oil Change Labor	Labor for standard oil change ser...
2 LABOR002	Brake Service Labor	Labor for brake pad replacement
3 LABOR003	Tire Service Labor	Labor for tire rotation and balanc...
4 LABOR004	Engine Service Labor	Labor for engine tune-up
5 LABOR005	Transmission Service Labor	Labor for transmission services
6 PART001	Oil Filter	Standard oil filter for most vehicles
7 PART002	Engine Oil (5W-30)	Synthetic oil 5W-30 grade, 1 quart
8 PART003	Brake Pads (Front)	Standard front brake pads set
9 PART004	Brake Pads (Rear)	Standard rear brake pads set
10 PART005	Air Filter	Standard air filter for most vehicles

Showing rows: 1 to 10 Page No: 1 of 1

✓ Successfully run. Total query runtime: 90 msec. 10 rows affected. X

pgAdmin 4

File Object Tools Edit View Window Help

public.invoice/Aut... X public.invoicecost... X public.invoiceesti... X public.lineitem/Au... X public.lineitemlabor/Automotive Repair Information System/postgres@PostgreSQL 17 X

Query History

```
1 ✓ SELECT * FROM public.lineitemlabor
2 ORDER BY itemno ASC
```

Data Output Messages Notifications

	Itemno [PK] character varying (10)	rate numeric (5,2)	hours numeric (5,2)
1	LABOR001	45.00	0.50
2	LABOR002	65.00	1.50
3	LABOR003	40.00	0.75
4	LABOR004	85.00	2.00
5	LABOR005	75.00	1.00

Showing rows: 1 to 5 | Page No: 1 of 1 | < << >> >

✓ Successfully run. Total query runtime: 79 msec. 5 rows affected. X

pgAdmin 4

File Object Tools Edit View Window Help

public.lineitempart/Automotive Repair Information System/postgres@PostgreSQL 17 X

Query History

```
1 ✓ SELECT * FROM public.lineitempart
2 ORDER BY itemno ASC
```

Data Output Messages Notifications

	Itemno [PK] character varying (10)	price numeric (7,2)
1	PART001	12.99
2	PART002	8.99
3	PART003	45.99
4	PART004	39.99
5	PART005	14.99

Showing rows: 1 to 5 | Page No: 1 of 1 | < << >> >

✓ Successfully run. Total query runtime: 70 msec. 5 rows affected. X

The screenshot shows the pgAdmin 4 interface with the following details:

- File Object Tools Edit View Window Help** menu bar.
- Connections**: Multiple connections are listed, including `cecost...`, `public.invoiceestl...`, `public.lineitem/Au...`, `public.lineitemlab...`, `public.lineitempart...`, and the current connection `public.listing/Automotive Repair Information System/postgres@PostgreSQL 17`.
- Toolbar**: Includes icons for New Connection, Open Connection, Save, Undo, Redo, Copy, Paste, Find, Replace, Refresh, and Help.
- Query History**: Shows the history of queries run, with the current one being the listing query.
- Scratch Pad**: A tab for temporary queries.
- Data Output**, **Messages**, and **Notifications** tabs at the bottom.
- Result Grid**: Displays the query results for the `listing` table. The columns are `invoiceno` (PK), `itemno` (PK), and `quantity`. The data consists of 12 rows.
- Status Bar**: Shows "Showing rows: 1 to 12" and "Page No: 1 of 1".
- Message Bar**: Shows a green success message: "Successfully run. Total query runtime: 73 msec. 12 rows affected."

	invoiceno	itemno	quantity
1	INV001	LABOR001	1
2	INV001	PART001	1
3	INV001	PART002	5
4	INV002	LABOR002	1
5	INV002	PART003	1
6	INV002	PART004	1
7	INV003	LABOR003	1
8	INV004	LABOR004	1
9	INV004	PART005	1
10	INV005	LABOR005	1
11	INV005	PART002	6
12	INV006	LABOR001	1

The screenshot shows the pgAdmin 4 interface with a query editor and a data output viewer.

Query Editor:

```
1 ✓ SELECT * FROM public.mechanic
2 ORDER BY staffno ASC
```

Data Output:

	staffno [PK] character varying (10)	fname character varying (50)	lname character varying (50)
1	MECH001	David	Jones
2	MECH002	Lisa	Thompson
3	MECH003	Carlos	Rodriguez
4	MECH004	Jessica	Miller
5	MECH005	Kevin	Anderson

Status Bar: Successfully run. Total query runtime: 83 msec. 5 rows affected.

pgAdmin 4

File Object Tools Edit View Window Help

public.procedure/Automotive Repair Information System/postgres@PostgreSQL 17

Query Query History

```
1 v SELECT * FROM public.procedure
2 ORDER BY procedureno ASC
```

Data Output Messages Notifications

procedureno	proceduredescription
PROC001	Oil Change
PROC002	Brake Pad Replacement
PROC003	Tire Rotation
PROC004	Engine Tune-Up
PROC005	Transmission Fluid Change

Showing rows: 1 to 5 Page No: 1 of 1

✓ Successfully run. Total query runtime: 71 msec. 5 rows affected. X

pgAdmin 4

File Object Tools Edit View Window Help

public.repairorder/Automotive Repair Information System/postgres@PostgreSQL 17

Query Query History

```
1 v SELECT * FROM public.repairorder
2 ORDER BY repairno ASC
```

Data Output Messages Notifications

repairno	originatedate	completedate	complaintnote	vin	staffno
REP001	2024-05-01	2024-05-01	Routine oil change and inspection	1HGCM82633A123456	MECH001
REP002	2024-05-05	2024-05-06	Squeaking brakes when stopping	5FNR38225B123456	MECH002
REP003	2024-05-10	2024-05-10	Tire wear uneven	2HGE165X9H123456	MECH003
REP004	2024-06-15	2024-06-17	Engine running rough	1FAHP2EW7CG123456	MECH001
REP005	2024-07-20	2024-07-20	Transmission shifting hard	WBADT63433C123456	MECH005
REP006	2024-08-01	2024-08-02	Check engine light on	1HGCM82633A123456	MECH004

Showing rows: 1 to 6 Page No: 1 of 1

✓ Successfully run. Total query runtime: 96 msec. 6 rows affected. X

pgAdmin 4

File Object Tools Edit View Window Help

public.requirement/Automotive Repair Information System/pos... public.repairorder... public.requirement/Automotive Repair Information System/postgres@PostgreSQL 17

Query Query History

```
1 ✓ SELECT * FROM public.requirement
2 ORDER BY procedurenno ASC, itemno ASC
```

Data Output Messages Notifications

	procedurenno	itemno	quantity
1	PROC001	LABOR001	1
2	PROC001	PART001	1
3	PROC001	PART002	5
4	PROC002	LABOR002	1
5	PROC002	PART003	1
6	PROC002	PART004	1
7	PROC003	LABOR003	1
8	PROC004	LABOR004	1
9	PROC004	PART005	1
10	PROC005	LABOR005	1
11	PROC005	PART002	6

Showing rows: 1 to 11 Page No: 1 of 1

✓ Successfully run. Total query runtime: 150 msec. 11 rows affected.

pgAdmin 4

File Object Tools Edit View Window Help

public.vehicle/Automotive Repair Information System/postgres@PostgreSQL 17

Query Query History

```
1 ✓ SELECT * FROM public.vehicle
2 ORDER BY vin ASC
```

Data Output Messages Notifications

	vin	make	year	model
1	1FAHP2EW7CG123456	Chevrolet	2021	Malibu
2	1HGCM82633A123456	Honda	2018	Accord
3	2HGES165X9H123456	Ford	2019	F-150
4	5FNRL38225B123456	Toyota	2020	Camry
5	WBADT63433C123456	BMW	2017	328i

Showing rows: 1 to 5 Page No: 1 of 1

✓ Successfully run. Total query runtime: 75 msec. 5 rows affected.

Case Study Application:

1. Insert a new customer into the database.
2. Delete an existing employee from the database.
3. Update the description of an existing repair order.
4. List all the repair orders belonging to a given vehicle, along with their dates when each repair was originated and completed.
5. List the details of the line items including the description, price, and quantity for the invoice(s) of a given repair order.
6. List the repair orders completed between June 2024 and December 2024, sorted by the repair order numbers.
7. List the details of all the line items of a given procedure.
8. List the total number of procedures required by each repair order in descending order, along with the procedure description.
9. List the name of the employee who recorded more than the average number of invoices, together with the number of invoices he/she recorded.
10. For a particular invoice, list the odometer mileages (in and out), payment information, and the vehicle information.

Sample Queries

```
-- 1. Insert a new customer into the database
INSERT INTO Vehicle (vin, make, year, model)
VALUES ('JH4KA7660PC123456', 'Acura', 2022, 'TLX');
-- Note: Changed to insert vehicle instead to reflect conceptual schema
```

pgAdmin 4

File Object Tools Edit View Window Help

Welcome Automotive Repair Information System/postgres@PostgreSQL 17* public.vehicle/Aut... X

Query History

```
1 -- 1. Insert a new customer into the database
2 ✓ INSERT INTO Vehicle (vin, make, year, model)
3   VALUES ('JH4KA7660PC123456', 'Acura', 2022, 'TLX');
4 -- Note: Changed to insert car instead to reflect conceptual schema
5
```

Data Output Messages Notifications

INSERT 1

Query returned successfully in 306 msec.

pgAdmin 4

File Object Tools Edit View Window Help

Welcome Automotive Repair... X public.vehicle/Automotive Repair Information System/postgres@PostgreSQL 17 X

Query History

```
1 ✓ SELECT * FROM public.vehicle
2 ORDER BY vin ASC
```

Data Output Messages Notifications

Showing rows: 1 to 6 Page No: 1 of 1

	vin	make	year	model
1	1FAHP2EW7CG123456	Chevrolet	2021	Malibu
2	1HGM82653A123456	Honda	2018	Accord
3	2HGES165X9H123456	Ford	2019	F-150
4	5FNL38225B123456	Toyota	2020	Camry
5	JH4KA7660PC123456	Acura	2022	TLX
6	WBADT63433C123456	BMW	2017	328i

Total rows: 6 Query complete 00:00:00.138 CRLF Ln 1, Col 1

-- 2. Delete an existing employee from the database. This only works if the mechanic doesn't have a repair order, so I will insert a new mechanic without and repair orders to show it being deleted.

-- Step 1: Insert a new mechanic with no associated repair orders.

```
INSERT INTO Mechanic (staffNo, fName, lName) VALUES ('MECH999', 'Jane', 'Smith');
```

-- Step 2: Verify the mechanic was added

```
SELECT * FROM Mechanic WHERE staffNo = 'MECH999';
```

The screenshot shows the pgAdmin 4 interface with the following details:

- File Bar:** File, Object, Tools, Edit, View, Window, Help.
- Toolbar:** Standard PostgreSQL toolbar with icons for connection, schema browser, table editor, etc.
- Query Editor:** Contains the following SQL code:

```
1 -- Step 1: Insert a new mechanic with no associated repair orders
2 ✓ INSERT INTO Mechanic (staffNo, fName, lName)
3   VALUES ('MECH999', 'Jane', 'Smith');
4
5 -- Step 2: Verify the mechanic was added
6 SELECT * FROM Mechanic WHERE staffNo = 'MECH999';
```
- Data Output:** Shows the results of the last query:

	staffno	fname	lName
1	MECH999	Jane	Smith

Showing rows: 1 to 1 | Page No: 1 | of 1 | < > | << >> | <<< >>> |
- Status Bar:** Total rows: 1 | Query complete 00:00:00.142 | CRLF | Ln 1, Col 66

pgAdmin 4

File Object Tools Edit View Window Help

Welcome public.mechanic/A... public.mechanic/Automotive Repair Information System/postgres@PostgreSQL 17

Query History

```
1 ✓ SELECT * FROM public.mechanic
2 ORDER BY staffno ASC
```

Data Output Messages Notifications

	staffno [PK] character varying (10)	fname character varying (50)	lname character varying (50)
1	MECH001	David	Jones
2	MECH002	Lisa	Thompson
3	MECH003	Carlos	Rodriguez
4	MECH004	Jessica	Miller
5	MECH005	Kevin	Anderson
6	MECH999	Jane	Smith

Showing rows: 1 to 6 | Page No: 1 of 1 |

Total rows: 6 Query complete 00:00:00.163 CRLF Ln 1, Col 1

-- Step 3: Delete the mechanic

DELETE FROM Mechanic WHERE staffNo = 'MECH999';

-- Step 4: Verify the mechanic was deleted

SELECT * FROM Mechanic WHERE staffNo = 'MECH999';

pgAdmin 4

File Object Tools Edit View Window Help

Welcome  Automotive Repair Information System/postgres@PostgreSQL 17*   

Query History  Scratch Pad 

```
-- Step 3: Delete the mechanic
DELETE FROM Mechanic
WHERE staffno = 'MECH999';
-- Step 4: Verify the mechanic was deleted
SELECT * FROM Mechanic WHERE staffNo = 'MECH999';
```

Data Output Messages Notifications 

staffno	fname	lname
---------	-------	-------

Total rows: 0 Query complete 00:00:00.108 CRLF Ln 6, Col 50

pgAdmin 4

File Object Tools Edit View Window Help

Welcome   

Query History  Scratch Pad 

```
SELECT * FROM public.mechanic
ORDER BY staffno ASC;
```

Data Output Messages Notifications 

staffno	fname	lname
MECH001	David	Jones
MECH002	Lisa	Thompson
MECH003	Carlos	Rodriguez
MECH004	Jessica	Miller
MECH005	Kevin	Anderson

Total rows: 5 Query complete 00:00:00.149 CRLF Ln 1, Col 1

```
-- 3. Update the description of an existing repair order
```

```
UPDATE RepairOrder
```

```
SET complaintNote = 'Squeaking brakes when stopping and vibration when applying brakes'
```

```
WHERE repairNo = 'REP002';
```

The screenshot shows the pgAdmin 4 interface with a single query window open. The query is:

```
1 -- 3. Update the description of an existing repair order
2 UPDATE RepairOrder
3 SET complaintNote = 'Squeaking brakes when stopping and vibration when applying brakes'
4 WHERE repairNo = 'REP002';
5
```

The 'Messages' tab at the bottom shows the results:

```
UPDATE 1
Query returned successfully in 64 msec.
```

At the bottom right, it says 'CRLF' and 'Ln 5, Col 1'.

pgAdmin 4

File Object Tools Edit View Window Help

Welcome

public.repairorder/Automotive Repair Information System/postgres@PostgreSQL 17

No limit ▾

Query Query History

```
1 ✓ SELECT * FROM public.repairorder
2 ORDER BY repairno ASC
```

Data Output Messages Notifications

Showing rows: 1 to 6 Page No: 1 of 1

	repairno [PK] character varying (10)	originatedate date	completedate date	complaintnote text	vin character varying (17)	staffno character varying (10)
1	REP001	2024-05-01	2024-05-01	Routine oil change and inspection	1HGCM82633A123456	MECH001
2	REP002	2024-05-05	2024-05-06	Squeaking brakes when stopping and vibration when applying brakes	5FNRL38225B123456	MECH002
3	REP003	2024-05-10	2024-05-10	Tire wear uneven	2HGES16SX9H123456	MECH003
4	REP004	2024-06-15	2024-06-17	Engine running rough	1FAHP2EW7CG123456	MECH001
5	REP005	2024-07-20	2024-07-20	Transmission shifting hard	WBADT63433C123456	MECH005
6	REP006	2024-08-01	2024-08-02	Check engine light on	1HGCM82633A123456	MECH004

Total rows: 6 Query complete 00:00:00.114 CRLF Ln 1, Col 1

-- 4. List all the repair orders belonging to a given vehicle

```
SELECT ro.repairNo, ro.originDate, ro.completeDate, ro.complaintNote
FROM RepairOrder ro
WHERE ro.vin = '1HGCM82633A123456'
ORDER BY ro.originDate DESC;
```

```

-- 4. List all the repair orders belonging to a given vehicle
SELECT ro.repairNo, ro.originDate, ro.completeDate, ro.complaintNote
FROM RepairOrder ro
WHERE ro.vin = '1HGCM82633A123456'
ORDER BY ro.originDate DESC;

```

Data Output

repairno	origindate	completedate	complaintnote
REP006	2024-08-01	2024-08-02	Check engine light on
REP001	2024-05-01	2024-05-01	Routine oil change and inspection

Total rows: 2 Query complete 00:00:00.268 CRLF Ln 6, Col 1

-- 5. List the details of the line items for the invoice(s) of a given repair order

SELECT

```

li.itemName,
li.description,
lip.price AS partPrice,
lil.rate AS laborRate,
lil.hours AS laborHours,
l.quantity
FROM Invoice i
JOIN Listing l ON i.invoiceNo = l.invoiceNo
JOIN LineItem li ON l.itemNo = li.itemNo
LEFT JOIN LineItemPart lip ON li.itemNo = lip.itemNo
LEFT JOIN LineItemLabor lil ON li.itemNo = lil.itemNo
WHERE i.repairNo = 'REP001'
ORDER BY li.itemName;

```

pgAdmin 4

File Object Tools Edit View Window Help

Welcome Automotive Repair Information System/postgres@PostgreSQL 17* public.mechanic/A... public.repairorder/...

Query History Scratch Pad

```

-- 5. List the details of the line items for the invoice(s) of a given repair order
SELECT
    li.itemName,
    li.description,
    lip.price AS partPrice,
    lil.rate AS laborRate,
    lil.hours AS laborHours,
    l.quantity
FROM Invoice i
JOIN Listing l ON i.invoiceNo = l.invoiceNo
JOIN LineItem li ON l.itemNo = li.itemNo
LEFT JOIN LineItemPart lip ON li.itemNo = lip.itemNo
LEFT JOIN LineItemLabor lil ON li.itemNo = lil.itemNo
WHERE i.repairNo = 'REP001'
ORDER BY li.itemName;

```

Data Output Messages Notifications

Showing rows: 1 to 3 Page No: 1 of 1 |◀|◀|▶|▶|

	itemname character varying (50)	description text	partprice numeric (7,2)	laborrate numeric (5,2)	laborhours numeric (5,2)	quantity integer
1	Engine Oil (5W-30)	Synthetic oil 5W-30 grade, 1 quart	8.99	[null]	[null]	5
2	Oil Filter	Standard oil filter for most vehicles	12.99	[null]	[null]	1
3	Standard Oil Change Labor	Labor for standard oil change service	[null]	45.00	0.50	1

Total rows: 3 Query complete 00:00:00.098 CRLF | Ln 16, Col 1

-- 6. List the repair orders completed between June 2024 and December 2024

```

SELECT repairNo, originDate, completeDate, complaintNote, vin, staffNo
FROM RepairOrder
WHERE completeDate BETWEEN '2024-06-01' AND '2024-12-31'
ORDER BY repairNo;

```

The screenshot shows the pgAdmin 4 interface. In the top-left corner, it says "pgAdmin 4". The menu bar includes "File", "Object", "Tools", "Edit", "View", "Window", and "Help". Below the menu is a toolbar with various icons. The main area has tabs: "Welcome" (selected), "Automotive Repair Information System/postgres@PostgreSQL 17*", "public.mechanic/A...", and "public.repairorder/...". The "Query" tab is active, displaying the following SQL code:

```

1 -- 6. List the repair orders completed between June 2024 and December 2024
2 SELECT repairNo, originDate, completeDate, complaintNote, vin, staffNo
3 FROM RepairOrder
4 WHERE completeDate BETWEEN '2024-06-01' AND '2024-12-31'
5 ORDER BY repairNo;
6

```

Below the code, the "Data Output" tab is selected, showing the results of the query:

	repairno	origindate	completedate	complaintnote	vin	staffno
1	REP004	2024-06-15	2024-06-17	Engine running rough	1FAHP2EW7CG123456	MECH001
2	REP005	2024-07-20	2024-07-20	Transmission shifting hard	WBADT63433C123456	MECH005
3	REP006	2024-08-01	2024-08-02	Check engine light on	1HGCM82633A123456	MECH004

At the bottom of the pgAdmin window, there is a status bar with "Total rows: 3" and "Query complete 00:00:00.153", along with "CRLF" and "Ln 6, Col 1".

-- 7. List the details of all the line items of a given procedure

SELECT

```

li.itemNo,
li.itemName,
li.description,
lip.price AS partPrice,
lil.rate AS laborRate,
lil.hours AS laborHours,
r.quantity

```

FROM Procedure p

JOIN Requirement r ON p.procedureNo = r.procedureNo

JOIN LineItem li ON r.itemNo = li.itemNo

LEFT JOIN LineItemPart lip ON li.itemNo = lip.itemNo

LEFT JOIN LineItemLabor lil ON li.itemNo = lil.itemNo

WHERE p.procedureNo = 'PROC001'

ORDER BY li.itemName;

```

-- 7. List the details of all the line items of a given procedure
SELECT
    li.itemNo,
    li.itemName,
    li.description,
    lip.price AS partPrice,
    lil.rate AS laborRate,
    lil.hours AS laborHours,
    r.quantity
FROM Procedure p
JOIN Requirement r ON p.procedureNo = r.procedureNo
JOIN LineItem li ON r.itemNo = li.itemNo
LEFT JOIN LineItemPart lip ON li.itemNo = lip.itemNo
LEFT JOIN LineItemLabor lil ON li.itemNo = lil.itemNo
WHERE p.procedureNo = 'PROC001'
ORDER BY li.itemName;

```

Data Output

itemno	itemname	description	partprice	laborrate	laborhours	quantity
PART002	Engine Oil (5W-30)	Synthetic oil 5W-30 grade, 1 quart	8.99	[null]	[null]	5
PART001	Oil Filter	Standard oil filter for most vehicles	12.99	[null]	[null]	1
LABOR001	Standard Oil Change Labor	Labor for standard oil change service	[null]	45.00	0.50	1

Total rows: 3 Query complete 00:00:00.100 CRLF Ln 17, Col 1

-- 8. List the total number of procedures required by each repair order

```

SELECT ro.repairNo, COUNT(d.procedureNo) AS procedureCount,
       STRING_AGG(p.procedureDescription, ', ') AS procedureDescriptions
  FROM RepairOrder ro
 JOIN Documentation d ON ro.repairNo = d.repairNo
 JOIN Procedure p ON d.procedureNo = p.procedureNo
 GROUP BY ro.repairNo
 ORDER BY procedureCount DESC;

```

The screenshot shows the pgAdmin 4 interface. In the top-left corner, it says "pgAdmin 4". The menu bar includes File, Object, Tools, Edit, View, Window, Help. The title bar shows "Welcome" and "Automotive Repair Information System/postgres@PostgreSQL 17*". Below the menu is a toolbar with various icons. The main area has two tabs: "Query" (selected) and "Scratch Pad". The "Query" tab contains the following SQL code:

```

1 -- 8. List the total number of procedures required by each repair order
2 SELECT ro.repairNo, COUNT(d.procedureNo) AS procedureCount,
3        STRING_AGG(p.procedureDescription, ', ') AS procedureDescriptions
4 FROM RepairOrder ro
5 JOIN Documentation d ON ro.repairNo = d.repairNo
6 JOIN Procedure p ON d.procedureNo = p.procedureNo
7 GROUP BY ro.repairNo
8 ORDER BY procedureCount DESC;
9

```

The "Data Output" tab shows the results of the query:

	repairno	procedurecount	proceduredescriptions
1	REP003	1	Tire Rotation
2	REP005	1	Transmission Fluid Change
3	REP004	1	Engine Tune-Up
4	REP001	1	Oil Change
5	REP002	1	Brake Pad Replacement

At the bottom, it says "Total rows: 5" and "Query complete 00:00:00.127".

-- 9. List mechanics who recorded more than the average number of invoices

```
SELECT m.fName, m.lName, COUNT(i.invoiceNo) AS invoiceCount
```

```
FROM Mechanic m
```

```
JOIN RepairOrder ro ON m.staffNo = ro.staffNo
```

```
JOIN Invoice i ON ro.repairNo = i.repairNo
```

```
GROUP BY m.staffNo, m.fName, m.lName
```

```
HAVING COUNT(i.invoiceNo) > (
```

```
    SELECT AVG(invoiceCount)
```

```
    FROM (
```

```
        SELECT COUNT(i2.invoiceNo) AS invoiceCount
```

```
        FROM Mechanic m2
```

```
        JOIN RepairOrder ro2 ON m2.staffNo = ro2.staffNo
```

```
        JOIN Invoice i2 ON ro2.repairNo = i2.repairNo
```

```
        GROUP BY m2.staffNo
```

```
    ) AS subquery
```

```
)
```

```
ORDER BY invoiceCount DESC;
```

The screenshot shows the pgAdmin 4 interface with a query window open. The query is a complex SQL statement involving multiple joins and subqueries to find mechanics who recorded more than the average number of invoices. The results are displayed in a table with three columns: fname, lname, and invoicecount. One row is shown for David Jones with a count of 2.

	fname	lname	invoicecount
1	David	Jones	2

-- 10. For a particular invoice, list the odometer mileages, payment and vehicle information
SELECT

```
i.invoiceNo,  
i.invoicePrintDate,  
i.odometerStart,  
ic.odometerFinish,  
ic.paymentDate,  
v.vin,  
v.make,  
v.model,  
v.year  
FROM Invoice i  
JOIN InvoiceCost ic ON i.invoiceNo = ic.invoiceNo  
JOIN RepairOrder ro ON i.repairNo = ro.repairNo
```

```
JOIN Vehicle v ON ro.vin = v.vin  
WHERE i.invoiceNo = 'INV001';
```

The screenshot shows the pgAdmin 4 interface with a query editor and a results grid.

Query Editor:

```
1 -- 10. For a particular invoice, list the odometer mileages, payment and vehicle information
2 SELECT
3     i.invoiceNo,
4     i.invoicePrintDate,
5     i.odometerStart,
6     ic.odometerFinish,
7     ic.paymentDate,
8     v.vin,
9     v.make,
10    v.model,
11    v.year
12 FROM Invoice i
13 JOIN InvoiceCost ic ON i.invoiceNo = ic.invoiceNo
14 JOIN RepairOrder ro ON i.repairNo = ro.repairNo
15 JOIN Vehicle v ON ro.vin = v.vin
16 WHERE i.invoiceNo = 'INV001';
17
```

Results Grid:

	invoiceno	invoiceprintdate	odometerstart	odometerfinish	paymentdate	vin	make	model	year
1	INV001	2024-05-01	35000	35010	2024-05-01	1HGCM82633A123456	Honda	Accord	2018

Total rows: 1 Query complete 00:00:00.126 CRLF Ln 17, Col 1

Conclusion

This report covered the logical data model and conceptual data model, with the EER diagrams for both of them included. There were also corrections and slight changes. Compared to the original logical and conceptual model, the following changes were made:

- Changed the data type for the “description” attribute in the **LineItem** table to “TEXT” instead of “INTEGER”.
- Changed the data type of the “invoicePrintDate” attribute for the **Invoice** table to “DATE” instead of “INTEGER”.
- Added a “quantity” attribute to the **Requirement** table.
- Corrected some of the Entity names to be the same as they are in the EER diagrams (Requirement and Documentation).
- Updated “vin” to be the primary key of the **Vehicle** entity and removed “vehicleNo” in the Logical data model.

The test data and SQL statements are included in the form of both typed documentation as well as screenshots. All tables were created and test data was inserted successfully. Instead of inserting a customer as the original instructions specified, a customer’s vehicle was inserted to align with the conceptual schema. The sample queries were completed successfully and are given in typed form and as screenshots. The sample output is also included in the screenshots.