

2025 CpSc H Q13

Section: Database Design and Development

Topic: Implementation Using SQL

Question Summary

Using the sample garage database (Customer, Vehicle, Part, Appointment), you must: (a) design a query to list full name, registration and part name for customer 'L Fletcher'; (b) write SQL to show the total value of parts in stock; (c) (i) find the average appointment cost, (ii) list appointments whose cost exceeds that average, in the shown order; (d) complete cardinalities when introducing an associative entity PartsListing.

Worked Solution

(a) Query design for customer 'L Fletcher'

Field(s) and calculation(s)	FullName: CONCAT(Customer.name) regNo: Vehicle.regNo partName: Part.partName
Table(s)	Customer, Vehicle, Appointment, Part
Search criteria	Customer.name = 'L Fletcher' AND Appointment.partID IS NOT NULL
Grouping	None
Sort order	regNo ASC, partName ASC (any sensible order accepted)

(b) SQL to display total value of all parts in stock

```
SELECT ROUND(SUM(quantity * price), 2) AS "Total value"
FROM Part;
```

Example output matches the sample: 4411.81

(c)(i) SQL to display average appointment cost

```
SELECT ROUND(AVG(cost), 2) AS "Average cost"
FROM Appointment;
```

Example output shown: 168.39

(c)(ii) SQL using saved query 'avgCost'

```
SELECT v.regNo, a.appID, v.make, v.model, v.year, a.cost
FROM Appointment a
JOIN Vehicle v ON v.regNo = a.regNo;
```

```
JOIN Vehicle v ON v.regNo = a.regNo
CROSS JOIN avgCost ac
WHERE a.cost > ac."Average cost"
ORDER BY a.cost DESC, v.regNo ASC;
```

This returns the rows in the order shown (highest cost first).

(d) Cardinalities with PartsListing

Appointment 1 — M PartsListing M — 1 Part

(PartsListing is an associative entity allowing multiple parts per appointment.)

Final Answer

(a) Fields: FullName (Customer.name), regNo, partName; Tables: Customer, Vehicle, Appointment, Part; Criteria: name = 'L Fletcher' (and partID IS NOT NULL); Grouping: none; Sort: regNo/partName.

(b) SELECT ROUND(SUM(quantity*price),2) AS "Total value" FROM Part;

(c)(i) SELECT ROUND(AVG(cost),2) AS "Average cost" FROM Appointment;

(c)(ii) SELECT v.regNo, a.appID, v.make, v.model, v.year, a.cost FROM Appointment a JOIN Vehicle v ON v.regNo=a.regNo CROSS JOIN avgCost ac WHERE a.cost > ac."Average cost" ORDER BY a.cost DESC;

(d) Appointment 1 — M PartsListing M — 1 Part.

Revision Tips

- Apply WHERE before ORDER BY; GROUP BY only when aggregating non-key columns.
- SUM(quantity*price) gives inventory value; use ROUND for presentation to 2 d.p.
- Many-to-many relationships are resolved with an associative entity that holds both FKs.

Customer

CustomerID
Name
Email
Phone

Vehicle

VehicleID
Make
Model

Appointment

AppointmentID
Date
CustomerID
VehicleID

Appointment

AppointmentID
Date
CustomerID
VehicleID

Part

PartID
Description
Price

