Assessment 2:

1. DONE
2. CREATE VIEW Task2 AS  
   SELECT \* FROM Car  
   ORDER BY make ASC, model ASC;
3. SELECT Ca.\*, Cu.\*, COUNT(J.CarId) AS CarJobAmount FROM Job J  
   INNER JOIN Car Ca on Ca.CarId = J.CarId  
   INNER JOIN Customer Cu ON Cu.CustomerID = J.CustomerId  
   GROUP BY Ca.CarId;
4. SELECT \* FROM Part  
   WHERE PartName  
   LIKE '%battery%'  
   AND partprice < '200';
5. SELECT J.\*, JC.JobCompletedTime FROM Job J  
   LEFT JOIN JobCompleted JC ON JC.JobId = J.JobId  
   WHERE JC.JobCompletedTime IS NULL;
6. CREATE VIEW Task6 AS  
   SELECT \*, (minutes / 60.0 \* 90) AS StaffTaskCost   
   FROM TaskStaff;
7. A.   
   CREATE VIEW Task7A AS  
   SELECT T.TaskId, IFNULL(SUM(P.PartPrice \* TP.Count), 0) as TotalPartsCost FROM Task T  
   LEFT JOIN TaskPart TP ON TP.TaskId = T.TaskId  
   LEFT JOIN Part P ON P.PartId = TP.PartId  
   GROUP BY T.TaskId;

B.  
CREATE VIEW Task7B AS  
SELECT IFNULL(SUM(T7.TotalPartsCost), 0) AS TotalPartsCost, J.JobId FROM Task7A T7  
LEFT JOIN Task T ON T.TaskId = T7.TaskId  
LEFT JOIN Job J ON J.JobId = T.JobId  
GROUP BY J.JobId;

8. CREATE VIEW Task8 AS  
SELECT T.JobId, IFNULL(SUM(T6.StaffTaskCost), 0) AS TotalLabourCost FROM Task6 T6  
LEFT JOIN Task T ON T.TaskId = T6.TaskId  
GROUP BY T.JobId;

1. **VIN-**

**Pros:**

* **guaranteed uniqueness from the VIN**
* **Already been made so don’t have to make it yourself**

Cons:

* **The number may be very long and take up more data storage than necessary**
* **Extra column needing to be made, means more data storage**

**UUID-**

**Pros:**

* **Choosing the identifier yourself so can be as flexible to cater to your own database needs**
* **Can choose shorter identifiers making easier for people to remember them if need be**
* **If using the SQL for own CarId, then it is guaranteed uniqueness, so no chance of duplication**

**Cons:**

* **If using a randomizer for ID creation, slight chance there can be 2 of the same**
* **Extra complexity with algorithms when it’s already been made with a VIN**
* **Extra column needing to be made, means more data storage**

1. CREATE VIEW Task10 AS   
   SELECT T7.JobId, T7.TotalPartsCost, T8.TotalLabourCost, (T7.TotalPartsCost + T8.TotalLabourCost) AS TotalCost  
   FROM Task7B T7  
   LEFT JOIN Task8 T8 ON T7.JobId = T8.JobId  
   ORDER BY TotalCost DESC;

11. CREATE VIEW Task11 AS

SELECT JC.JobId, IFNULL(SUM(P.Amount), 0) AS AmountPaid,

IFNULL(MAX(P.timestamp), JC.JobCompletedTime) AS LatestTimestamp,

COUNT(P.JobId) AS NumberOfPayments FROM Payment P

RIGHT JOIN JobCompleted JC ON P.JobId = JC.JobId

GROUP BY P.JobId;

12: A.

CREATE VIEW Task12A AS

SELECT T10.JobId, (T10.TotalCost - T11.AmountPaid) AS BalanceOutstanding,

T11.NumberOfPayments, T11.LatestTimestamp AS LastPaymentTimestamp

FROM Task11 T11

LEFT JOIN Task10 T10 ON T11.JobId = T10.JobId;

B.

CREATE VIEW Task12B AS

SELECT C.CustomerId, C.FirstName, C.LastName, SUM(T12A.BalanceOutstanding) AS BalanceOutstanding,

SUM(T12A.NumberOfPayments) AS NumberOfPayments,

T12A.LastPaymentTimestamp, MAX(strftime('%d-%m-%Y', LastPaymentTimestamp, 'unixepoch')) AS LastPaymentDate

FROM Customer C

LEFT JOIN Job J ON C.CustomerID = J.CustomerId

RIGHT JOIN Task12A T12A ON J.JobId = T12A.JobId

GROUP BY C.CustomerID;

C.

CREATE VIEW Task12C AS

SELECT \* FROM Task12B

WHERE BalanceOutstanding > 0

AND (1476057600 - LastPaymentTimestamp) > (45 \* 24 \* 60 \* 60);