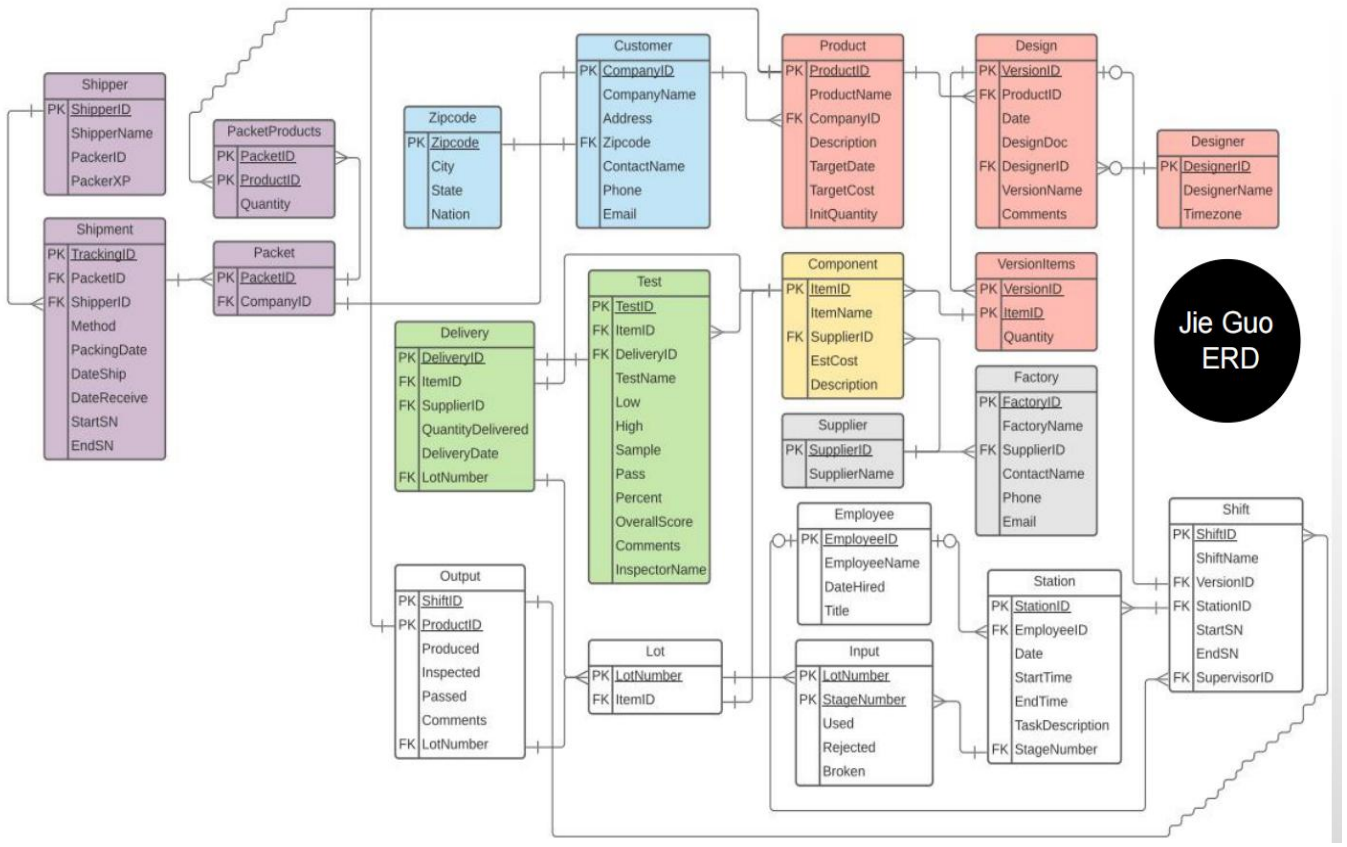
SQLocity: Database Project

Jie Guo 989354956

Relational Schema (as ERD):



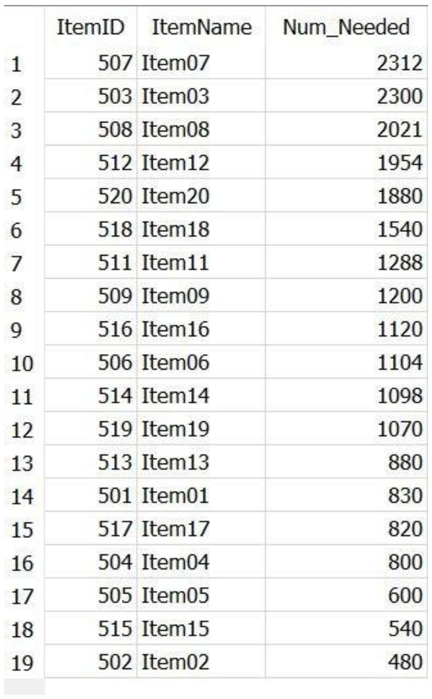
All our SQL calculations were made with SQLite, and the ERD was reproduced using LucidChart.io

Query #1

Which component is most needed across products? Order the components required in descending order.

```
SELECT
        VersionItems.ItemID,
        Component.ItemName,
        sum(Product.InitQuantity*VersionItems.Quantity) as Num_Needed
FROM
        Product join Design on Product.ProductID = Design.ProductID
        join VersionItems on Design.VersionID = VersionItems.VersionID
        join Component on Component.ItemID =VersionItems.ItemID
GROUP BY
        Component.ItemID
HAVING
        Design.Date = max(Design.Date)
ORDER BY
        Num_Needed DESC
```

OUTPUT:



	ItemID	ItemName	Num_Needed
1	507	Item07	2312
2	503	Item03	2300
3	508	Item08	2021
4	512	Item12	1954
5	520	Item20	1880
6	518	Item18	1540
7	511	Item11	1288
8	509	Item09	1200
9	516	Item16	1120
10	506	Item06	1104
11	514	Item14	1098
12	519	Item19	1070
13	513	Item13	880
14	501	Item01	830
15	517	Item17	820
16	504	Item04	800
17	505	Item05	600
18	515	Item15	540
19	502	Item02	480

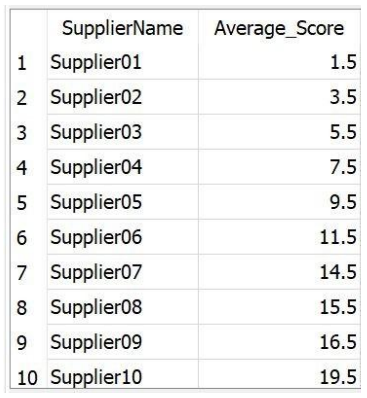
In order to find the most needed component across products, we performed a Groupby on the Component's ItemID. For this to make sense, we had to join several tables. We joined our way from Product to Component tables, through Design and VersionItems. A product has many versions, and each version has a list of items (listed in VersionItems) with its own quantity listed. Product versions are sequential (the most recently created is an update of the previous), so we only selected those versions where the Design.Date was the maximum. For each Product, the Customer specified an Initial Quantity that they would request. If you multiply this by the quantity of each Component, you get the total number of required components (of a particular type) for an entire order. To do this across all products, we did a sumproduct of InitQuantity * VersionItemsQuantity. Finally, we ordered the results in descending order. The most needed component is Item07.

Query #2

Which supplier has the lowest quality? Use overall score average.

```
SELECT
            Supplier.SupplierName,
            AVG(Test.OverallScore) as Average_Score
FROM
            Supplier join Component on Component.SupplierID = Supplier.SupplierID
            join Test on Component.ItemID =Test.ItemID
GROUP BY
            Supplier.SupplierID
ORDER BY
            Test.OverallScore ASC
```

OUTPUT



	SupplierName	Average_Score
1	Supplier01	1.5
2	Supplier02	3.5
3	Supplier03	5.5
4	Supplier04	7.5
5	Supplier05	9.5
6	Supplier06	11.5
7	Supplier07	14.5
8	Supplier08	15.5
9	Supplier09	16.5
10	Supplier10	19.5

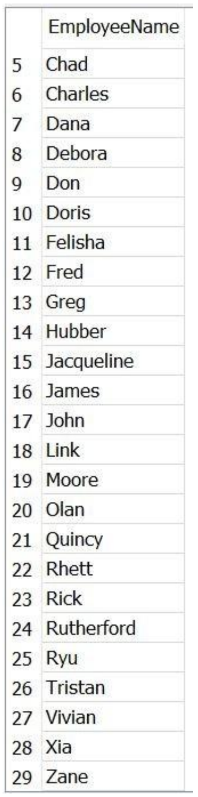
To find the average score of a Supplier's Items, we joined Supplier to Component to Test (where the OverallScore resides). Performing a Groupby on Supplier, we averaged each supplier's OverallScore (for all items they sold), and ordered by Ascending. Supplier01 had the lowest quality.

Query #3

Identify employees that are not assigned to a production stage.

```
SELECT
        Employee.EmployeeName
FROM
        Employee
EXCEPT
SELECT
        Employee.EmployeeName
FROM
        Employee join Station on Employee.EmployeeID = Station.EmployeeID
```

OUTPUT:



	EmployeeName
5	Chad
6	Charles
7	Dana
8	Debora
9	Don
10	Doris
11	Felisha
12	Fred
13	Greg
14	Hubber
15	Jacqueline
16	James
17	John
18	Link
19	Moore
20	Olan
21	Quincy
22	Rhett
23	Rick
24	Rutherford
25	Ryu
26	Tristan
27	Vivian
28	Xia
29	Zane

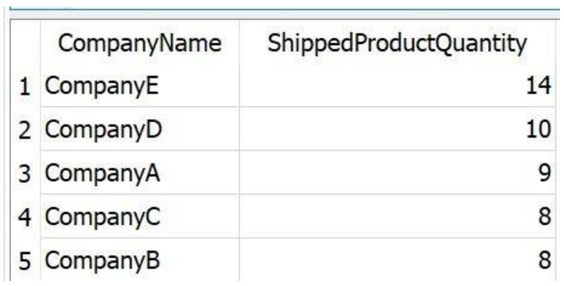
Since an employee cannot be assigned to a production stage (i.e. have a StageNumber) without being on a Station, we are looking for employees whose EmployeeID's are *not* in the Station table. To solve this problem, all we needed was a simple set minus. Using EXCEPT, we subtracted the set of *all employees* from the set of *employees with a production stage*. The query takes Employees from the Employee table and EXCEPTs theEmployeeID's of the join of the Employee table and the Station table. In total, there are 29 employees who are not assigned to a production stage, out of 39 employees. 10 of those are supervisors (which oversee shifts, not production stages), 1 is a janitor, and the other 9 are unassigned workers.

Query #4

Who is the biggest customer by item in terms of shipped product quantity?

```
SELECT
        Customer.CompanyName,
        Sum(PacketProducts.Quantity) as ShippedProductQuantity
FROM
        Customer join Packet on Customer.CompanyID = Packet.CompanyID
        join PacketProducts on PacketProducts.PacketID =Packet.PacketID
GROUP BY
        Customer.CompanyID
ORDER BY
        ShippedProductQuantity DESC
```

OUTPUT:



	CompanyName	ShippedProductQuantity
1	CompanyE	14
2	CompanyD	10
3	CompanyA	9
4	CompanyC	8
5	CompanyB	8

To find the biggest Customer, we performed a Groupby on CompanyID (we work with companies). Since the question pertains to shipped product quantity, we joined the tables Customer, Packet, and PacketProducts, the latter of which contained the quantity of the particular products we were interested in. We summed over each CompanyID, adding their quantities, and ordered by descending. Our biggest customer is CompanyE, at 14 shipped products.