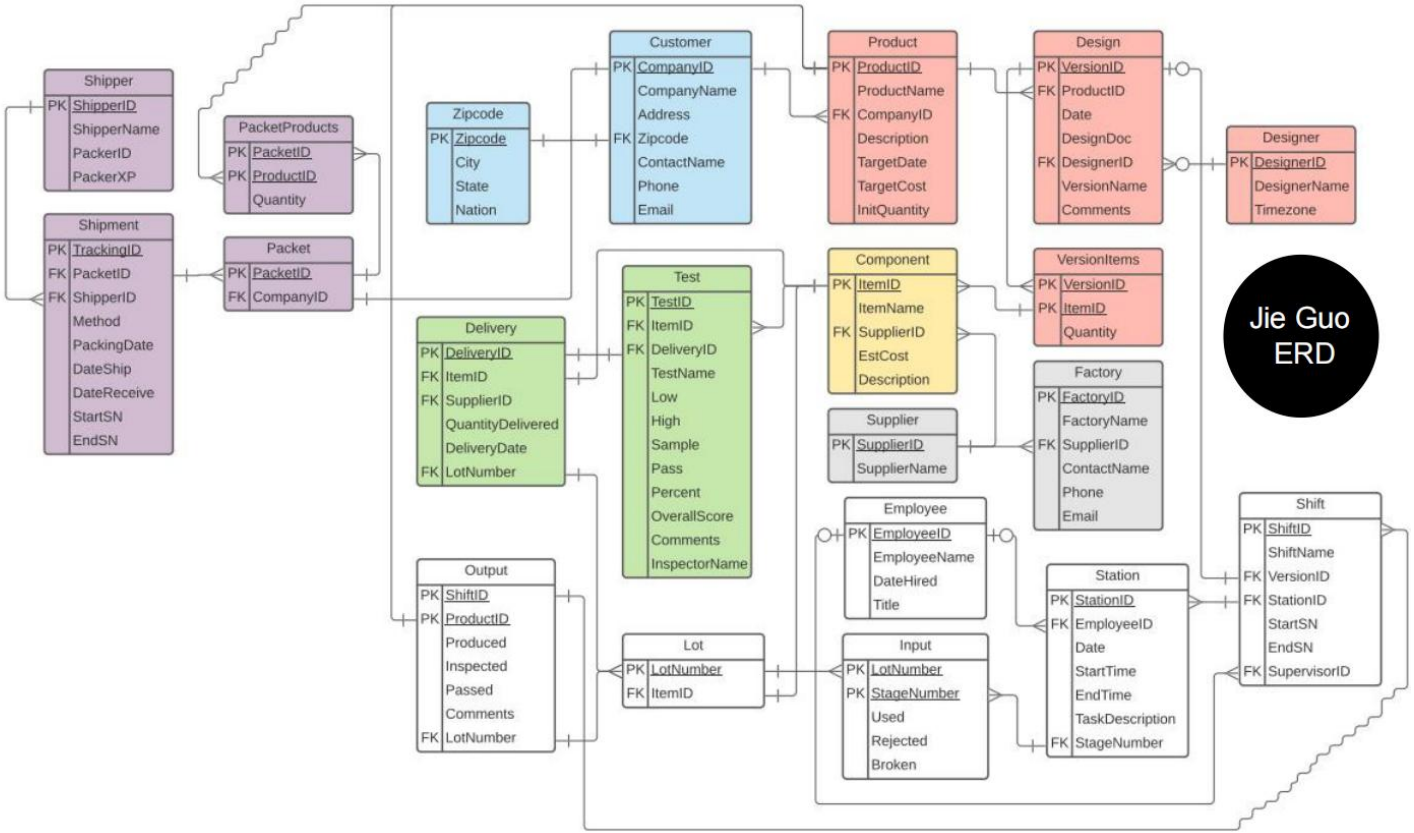
# SQLocity

Database Construction

Jie Guo 989354956

# Work Contents

DB Schema design: Designed tables, added cardinalities, provided keys, decided on structure.
DB Implementation: Recreated tables in database
SQL Queries: Solved query logic
Data Generation:  Generated fake data for our database


Shipper
PK ShipperID
ShipperName
PackerID
PackerXP
PacketProducts
PK PacketID
PK ProductID
Quantity
Zipcode
PK Zipcode
City
State
Nation
Customer
PK CompanyID
CompanyName
Address
FK Zipcode
ContactName
Phone
Email
Product
PK ProductID
ProductName
FK CompanyID
Description
TargetDate
TargetCost
InitQuantity
Design
PK VersionID
FK ProductID
Date
DesignDoc
FK DesignerID
VersionName
Comments
Designer
PK DesignerID
DesignerName
Timezone
Shipment
PK TrackingID
FK PacketID
FK ShipperID
Method
PackingDate
DateShip
DateReceive
StartSN
EndSN
Packet
PK PacketID
FK CompanyID
Test
PK TestID
FK ItemID
FK DeliveryID
TestName
Low
High
Sample
Pass
Percent
OverallScore
Comments
InspectorName
Component
PK ItemID
ItemName
FK SupplierID
EstCost
Description
VersionItems
PK VersionID
PK ItemID
Quantity
Jie Guo
ERD
Delivery
PK DeliveryID
FK ItemID
FK SupplierID
QuantityDelivered
DeliveryDate
FK LotNumber
Supplier
PK SupplierID
SupplierName
Factory
PK FactoryID
FactoryName
FK SupplierID
ContactName
Phone
Email
Employee
PK EmployeeID
EmployeeName
DateHired
Title
Shift
PK ShiftID
ShiftName
FK VersionID
FK StationID
StartSN
EndSN
FK SupervisorID
Output
PK ShiftID
PK ProductID
Produced
Inspected
Passed
Comments
FK LotNumber
Lot
PK LotNumber
FK ItemID
Input
PK LotNumber
PK StageNumber
Used
Rejected
Broken
Station
PK StationID
FK EmployeeID
Date
StartTime
EndTime
TaskDescription
FK StageNumber

# Schema

# SQLite

DB Browser for SQLite Version 3.12.1
SQLite database with 21 tables

# Query #1

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

```sql
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
```

| | 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, I had to join several tables. I joined our way from Product to Component tables, through Design and VersionItems. A product has many versions (VersionID), 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 Item, 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, I ordered the results in descending order. The most needed component is Item07.

# Query #2

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

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

```
1  SELECT
2    Supplier.SupplierName,
3    AVG(Test.OverallScore) as Average_Score
4  FROM
5    Supplier join Component on Component.SupplierID = Supplier.SupplierID
6    join Test on Component.ItemID = Test.ItemID
7  Group by
8    Supplier.SupplierID
9  Order by
10   Test.OverallScore ASC
```

| | 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 |

# Query #3

Identify employees that are not assigned to a production stage.

```
1   SELECT
2       Employee.EmployeeName
3   FROM
4       Employee
5   EXCEPT
6   SELECT
7       Employee.EmployeeName
8   FROM
9       Employee join Station on Employee.EmployeeID = Station.EmployeeID;
```

| | 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 I 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 EmployeeID's from the Employee table and EXCEPTs EmployeeID'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?

```
1   SELECT
2       Customer.CompanyName,
3       Sum(PacketProducts.Quantity) as ShippedProductQuantity
4   FROM
5       Customer join Packet on Customer.CompanyID = Packet.CompanyID
6       join PacketProducts on PacketProducts.PacketID = Packet.PacketID
7   Group by
8       Customer.CompanyID
9   Order by
10      ShippedProductQuantity DESC
```

| | 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 I was interested in. We grouped over each CompanyID, summing their Quantities, and ordered by descending. Our biggest customer is CompanyE, at 14 shipped products.