**Part 1 Triggers**

1. **Each delivery request has at least one package**Most likely **initially deferrable –** circular dependency?: In the case where the transaction starts with inserting the delivery requests first, THEN inserts the packages. Else will immediately throw exception.  
     
   For each delivery\_request inserted into table, check in packages table if there exists at least one package with packages(request\_id) = delivery\_request(id)  
   Need to handle case where cannot find = Return NULL  
   Probably throw exception if the delivery\_requests inserted does not have a package to terminate the transaction.  
     
   *Check for empty/null table – No delivery requests at all*
2. **For each delivery request, the IDs of the packages should be consecutive integers starting from 1.**On packages table, group by request\_id, then order by increasing  
   Can run a cursor through the tables. Declare a variable in the function to keep track of the next expected ID.  
   Whenever the request\_id is changed, we will reset everything – reset to 1, current\_id = new request\_id  
     
   *Check for empty/null table – Packages table is empty, NULL values*
3. **For each delivery request, the IDs of the unsuccessful pickups should be consecutive integers starting from 1.**Similar to the one above, but on unsuccessful\_pickups table.  
   Group by request\_id, order ascending for pickup\_id.  
   Reset when request\_id changes.  
     
   *Check for empty/null table – Unsuccessful pickups is empty*
4. **Timestamp of first unsuccessful pickup should be after submission\_time of the corresponding delivery request.** **Each further unsuccessful pickup timestamp should be after previous unsuccessful pickup’s timestamp (IF ANY)**Need take note of case where there is no previous timestamp (NULL)  
   Also need to check for possible pickups with no unsuccessful.  
   If only 1 unsuccessful, just need to check against submission only.  
     
   Sort unsuccessful pickups by request\_id  
     
   unsuccessful\_pickups(request\_id) link to delivery\_requests(id)  
   IF statement for submission\_time and pickup\_time – Throw exception if required  
   Next, need to cursor – Order by ascending pickup\_id and check the pickup\_time of each is in ascending order

**Legs Related**

1. **For each delivery request, the IDs of the legs should be consecutive integers starting from 1.**Similar, group by request\_id, sort by leg\_id ascending, check if consecutive
2. **Start time of first leg of each delivery request should be after submission\_time of the delivery request and the timestamp of the last unsuccessful pickup (IF ANY)**  
   Special consideration that the first leg HAS NOT BEGUN YET (no first leg for delivery request)  
     
   Need to check if unsuccessful pickup has occurred before.   
   From legs table, leg\_id = 1  
   Check request\_id in unsuccessful\_pickups, pickup\_time
3. **For each delivery request, new leg cannot be inserted if start\_time is before end\_time of previous leg, or end\_time of previous leg is NULL**Need special case, cause if new delivery request and is first to be inserted  
   Check if previous legs exist (count > 1), else just insert  
   If previous does exist, do the NULL and timing check  
   Make sure delivery request exists when inserting leg also?
4. **Timestamp of each unsuccessful\_delivery should be after the start\_time of corresponding leg.**Group all unsuccessful\_deliveries together by request\_id.   
   Determine leg\_id start\_time unsuccessful is attributed with.  
   Ensure all timestamps in unsuccessful\_deliveries are after the start\_time of leg.
5. **Each delivery request, at most 3 unsuccessful\_deliveries**Find all delivery request IDs  
   Need to handle cases where request ID has no unsuccessful delivery COUNT(\*) = NULL  
   Probably use coalesce  
   After insertion of unsuccessful\_delivery?
6. **Cancel\_time of cancelled request after submission\_time of corresponding delivery request.**Ensure id in cancelled\_request exists in delivery\_requests  
   Just check cancel\_time > delivery\_request
7. **For each delivery request, return\_leg ID should be 1, 2 consecutive**Check for when delivery request has no return\_legs (Grouping = empty)  
   Order group by leg\_id  
   Ensure ensure start\_time of subsequent leg after end time of previous
8. **Delivery request – First return-leg CANNOT BE INSERTED if no existing leg for delivery request OR last existing leg’s end\_time time is after start\_time of return\_leg AND return\_leg start\_time should be after cancel\_time of request (IF ANY)**On insert of return\_leg – check legs table if any leg exists  
   Also check if end\_time is > start\_time of return leg (invalid insert)  
   AND check cancel\_time is < return\_leg Start\_time
9. **Each delivery request – At most 3 unsuccessful\_return\_deliveries**On insert of unsuccessful\_return\_deliveries  
   Sum all current ones. If = 3, reject  
   Special case if count = 0 (ensure not NULL) – First unsuccessful return delivery
10. **Timestamp of each unsuccessful\_return\_delivery should be after the start\_time of the coreresponding return\_leg**Match record in unsuccessful\_return\_deliveries to return\_legs by request\_id and leg\_id  
    Possible that record does not exist in return\_legs  
    Check attempt\_time > start\_time return\_legs

**FOR ALL – DO WE NEED TO CHECK IF IS IN ACCEPTED/CANCELLED FIRST? e.g ONLY CANCELLED CAN BE RETURN LEG, ONLY ACCEPTED HAVE LEGS**