**Requirement #1:**

1. **1NF Dependency Chart:**

From the provided reports I was able to extrapolate 17 core attributes that would allow me to derive the final 3NF tables. I did not choose to the Reserved Seats or the Seats Remaining as core attributes for the following reasons.

* 1. The reserved seats and seats remaining, I would be dynamic attributes which during class we were told we would not be responsible for. This led me to the conclusion that they would be dynamically created using reports and subqueries.
  2. My reasoning for the chosen PK is by using the given business rule that any flight can be uniquely identified by the **Flight\_Nbr** and the **Flight\_Date** all I had to do was find at least one additional attribute to uniquely identify a given entity instance in the 1NF table.
     1. **Cust\_Nbr:** I chose this as the additional attribute since each flight would only have one instance of a given customer entity.

1. **3NF Dependency Chart:**

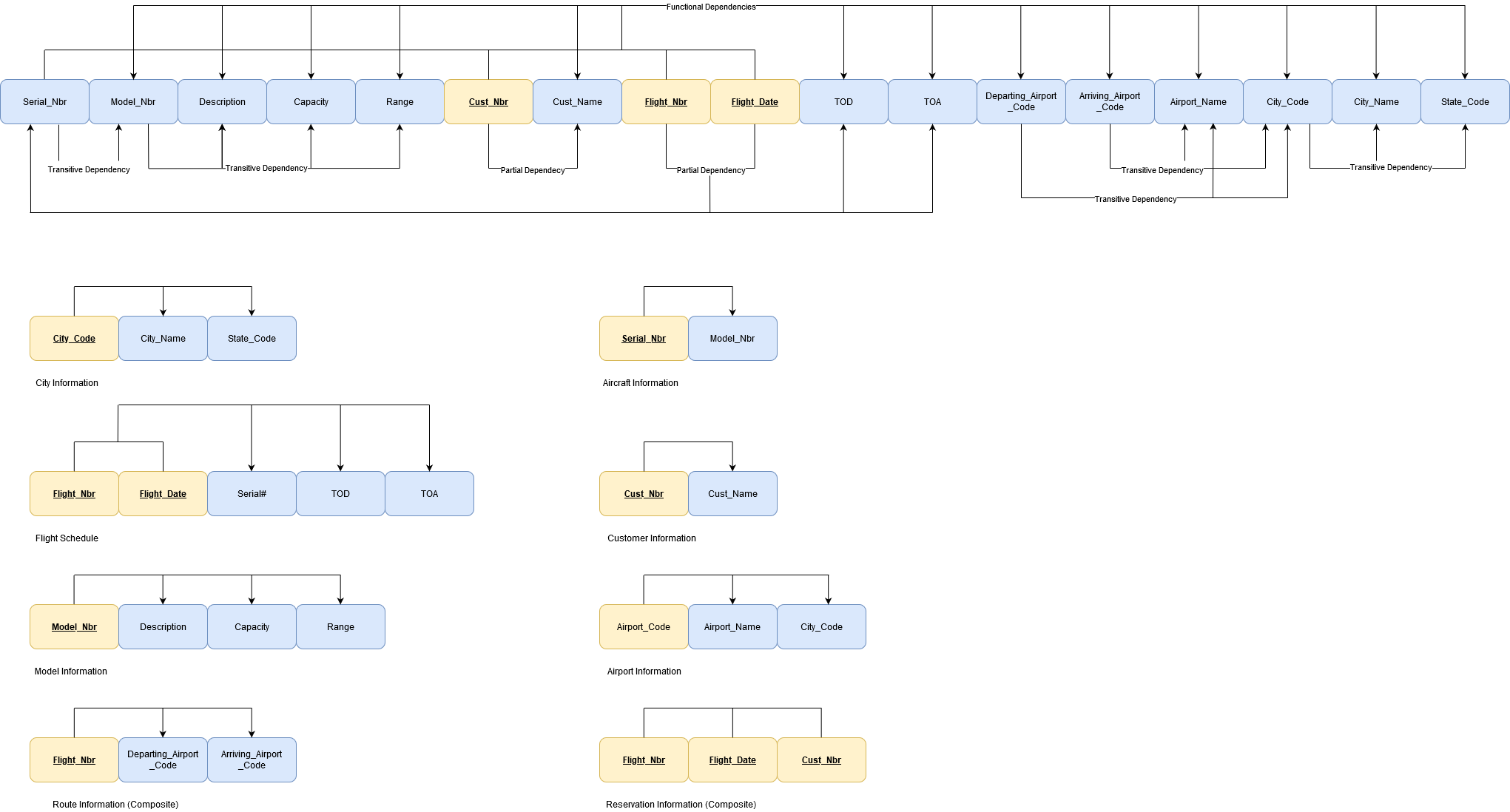
I was able to work my way down to 3NF by making sure the proper transitive and partial dependencies were revealed in the 1NF chart. I worked my way through a 2NF chart to check the results of the 3NF chart.

* 1. During the 2NF phase I saw the necessity for a third attribute (Airport\_Code) which would act as the primary key in its own table. This would allow the Flight Number table to be able to reference it for its departure and arrival airport codes. This would also allow for the minimizing of redundant data.

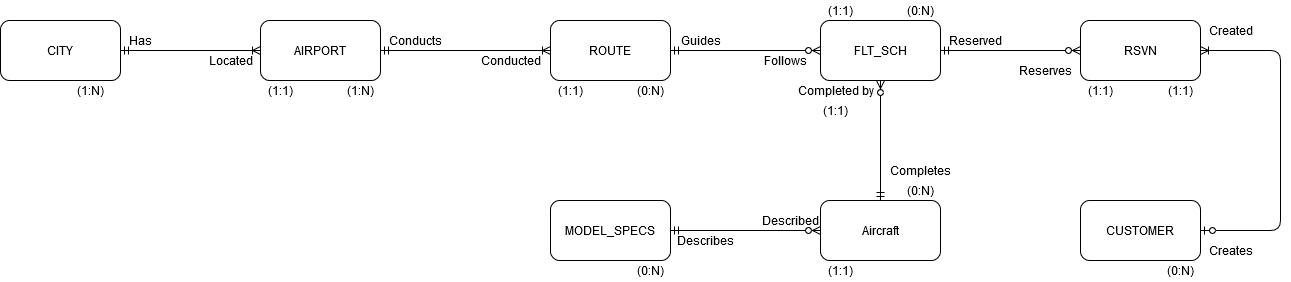
1. **Thoughts:**

This was very hard to do. It will probably be the hardest part of the project in my opinion. I had the most trouble deciding between how many attributes to include in the 1NF for multiple reasons.

* 1. The first problem I arrived at was whether to have the Airport\_Code in the 1NF chart. I decided against it as it seemed like a redundant group which would violate the 1NF rules. At least, that is my conclusion.
  2. The second problem I created for myself was the PK. I was overthinking it at first and making wild cases where it would be possible for a customer to take the same flight in one day. I justified this by thinking of the flight from Dallas to Austin. It is very possible to take that flight twice in one day, however ultimately it is a little farfetched, so I decided against using the **Serial\_Nbr, TOD** or the **TOA** as part of the primary key.
  3. I have also realized that I may have a bit of a shortcoming with my understanding on how to show transitive or partial dependencies best/properly. As in, I do not really know if an attribute can be part of multiple dependencies even when the dependency is different by just one attribute. I ultimately decided with the approach I took after conferring with others and doing some research.



1. **ER-Diagram:**



1. **ER-Thoughts: (From left to right)**
   1. City: Each city can have more than one airport, but it must have at least one to be recorded into the database. Otherwise there could be lots of “useless” data in the table.
   2. Airport: Each airport has only one city where it is located and can exist in more than one but must exist in at least one. Otherwise it would be a deprecated airport with no need for it to be recorded.
   3. Route: This one was tricky, as it technically needs two airports to fulfill the arrival and departure aspect of its data. However, I could not find a way to show this property. I ultimately decided on a 1:1 because a route must arrive or depart from one airport. And cannot technically arrive or depart from more than one. Especially with the simplified business we were given with no connecting flights.
   4. Flight Schedule: Each scheduled flight must have one and only one route to follow. Likewise, one and only one aircraft completes that scheduled flight. After talking out how real world airlines work in that a flight is posted without anyone having booked it at first I decided on a 0:N relationship in regards to reservations. It would not be till closer to the day of the flight when an airline decides to drop/merge the flight if not enough seats are booked creating a large loss in revenue.
   5. Aircraft: An aircraft may exist, but not be scheduled to fly on a given day for many reasons, such as maintenance, but it can be scheduled for more than one flight if the route scheduling allows it. If it exists than it was created using predefined blueprints of a particular model which is why it has a 1:1 relationship with Model Specs.
   6. Model Specs: A blueprint can be created, but not have any built aircrafts at the time of its realization. As in, the plans for a particular model can exist without an aircraft based off the plans existing as well.
   7. Reservation: Each booking applies to one and only one scheduled flight. If another flight is needed, then there would be another booking for the other scheduled flight. Each booking also applies to one and only one customer. As in, no two customers can have the same reservation.
   8. Customer: Each customer can have more than one reservation, but also does not necessarily need to have one if they are not flying on a given day. Customer info would be retained to allow for speedier booking in the future resulting in a 0:N relationship.
2. **Relational Schema:**
   1. CITY (**City\_Code**, City\_Name, State\_Code)
      1. PK: City\_Code
   2. AIRPORT (**Airport\_Code,** City\_Code, Airport\_Name)
      1. PK: Airport\_Code
      2. FK: City\_Code
   3. ROUTE (**Flight\_Nbr,** DP\_ARPT\_Code, AR\_ARPT\_Code)
      1. PK: Flight\_Nbr
      2. FK: DP\_ARPT\_Code, AR\_ARPT\_Code
   4. FLT\_SCH (**Flight\_Nbr, Flight\_Date**, Serial\_Nbr, TOD, TOA)
      1. PK: (Flight\_Nbr, Flight\_Date)
      2. FK: Flight\_Nbr, Serial\_Nbr
   5. RSVN (**Cust\_Nbr, Flight\_Nbr, Flight\_Date**)
      1. PK: (Cust\_Nbr, Flight\_Nbr, Flight\_Date)
      2. FK: Cust\_Nbr, Flight\_Nbr, Flight\_Date
   6. CUSTOMER (**Cust\_Nbr,** Cust\_Name)
      1. PK: Cust\_Nbr
   7. AIRCRAFT (**Serial\_Nbr,** Model\_Nbr)
      1. PK: Serial\_Nbr
      2. FK: Model\_Nbr
   8. MODEL\_SPECS (**Model\_Nbr**, Description, Capacity, Range)
      1. PK: Model\_Nbr
3. **Data Dictionary:**

