**(Outline, to be fleshed out)**

**Describe Constraints**

* There must be at least one engine-qualified employee per train voyage
* Newly-scheduled voyages must occur in the future
* Pretty much every number we use must be positive (price, capacity, years, distance)
* The number of tickets for a passenger car on a voyage must not exceed the number of seats on that passenger car
* A ticketholder cannot own multiple tickets for the same voyage (that person’s name may only be on one ticket)
* A ticketholder cannot acquire tickets for voyages that began in the past
* On a given voyage, there must be baggage cars with enough capacity to store the baggage for all ticketholders
* On a given voyage, there must be dining cars with enough capacity to satiate all ticketholders
* On a given voyage, there must be exactly one engine
* On a given voyage, there cannot be more cars than the engine can pull
* An engine type’s date of invention cannot be in the future
* A voyage’s arrival time must be after its departure time
* No two voyages starting at the same time can begin on the same track
* All engines and cars currently on a voyage must be in service
* A train route must be continuous and may not visit the same station twice (otherwise direction will become arbitrary)

**Describe Design Rationale/Choices/Reasoning**

* Rather than each voyage following a unique route, we chose to have voyages travel along set routes
  + This makes our database slightly less flexible (in order to create a Train Voyage along a new route, we must first create a Train Route entry)
  + However, this duplication in the case where multiple Train Voyages follow the same Train Route
  + This is also more realistic, as real-life trains follow a limited, predictable pattern
* Tickets, not Passengers, have a are directly related to a Passenger Car
  + Although it might seem rational to have a direct relationship between Passenger and Passenger Car, it is simpler to have an intermediate ticket
  + The Ticket is concerned with the Voyage ID and Passenger Car ID, the Passenger need only be concerned with the information printed on the Ticket
  + This allows us to compartmentalize what does change
* Train Routes are organized as a sets of track sections between stations.
  + This is equivalent to describing a path through a graph by listing an ordered set of edges
  + Our voyage length is not fixed
  + Our voyage is able to contain cycles and cross over itself
  + Disadvantage: a Track Section must appear as a tuple once for every time it is traversed in every track, meaning the Track Section/Train Route Pair relation might be very large
* Cars are related directly to a Voyage, not to the Engine that is pulling them
  + This way, a Car doesn’t care what train is pulling it, saving us the trouble of reassigning cars and engines for each voyage
  + This is a slight departure from reality, since our model essentially permits instantaneous car coupling/decoupling between two Voyages
  + However, relating a Car to an Engine rather than a Voyage would make querying Cars by Voyage more costly and complex