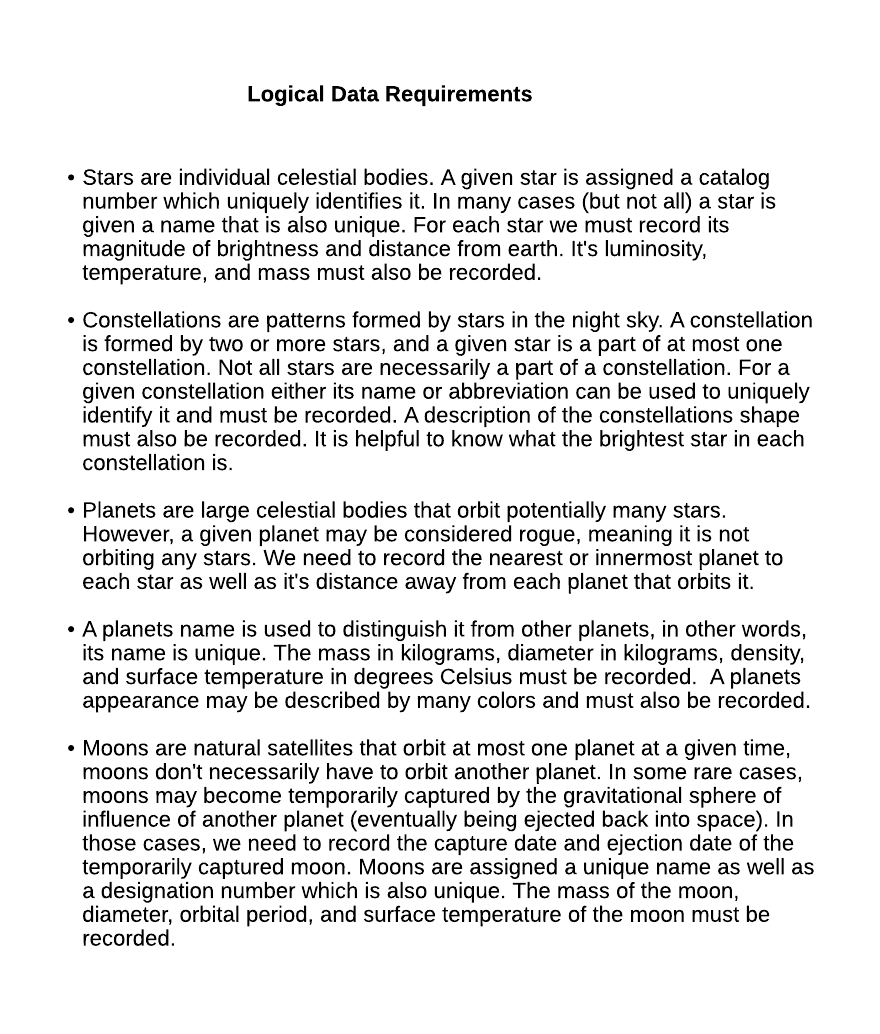
**Step 1 – Requirements Specification & Analysis (this is done for you).**

* An end user must be able to see a list of all constellations with the name and catalog number of the stars that make them up.
* Having selected a specific star (by catalog number) the end user should be able to view the nearest planet that is orbiting that star, as well as other planets orbiting that star.
* The end user should be able to create a moon.
* The end user should be able to eject a moon from the orbit of a planet.
* The end user should be able to delete a planet (for example Pluto) along with all the moons that are orbiting it, or that have been captured by it.

**Step 2 – Conceptual Model**

Produce the *ER diagram* corresponding to the *Logical Data Requirements* (above).

**ER Diagram:**

**Step 3 – Logical Model**

Map the *ER Diagram* to the *Relational Schema* using the mapping procedure outlined in Chapter 9 of the textbook (note any assumptions made during the mapping process, be sure to define a domain for each attribute in the completed relational schema).

**Relational Schema:**

**Step 4 – Implementation Model**

Implement each *Relation* from the *Relational Schema* as a *SQL table* (be sure to include domain, key, entity integrity, and referential integrity constraints where applicable).

**SQL Script(s) (to create tables):**

Seed each SQL Table with 5 – 10 tuples of data.

**SQL Script(s) (to seed each table):**

**Step 5 – Physical Model**

Create endpoints in a Java Spring Boot API to fulfill the functional requirements of the system (above). Demonstrate the fulfillment of each functional requirements using Postman.