Database Systems

PROJECT TITLE : SPACE DEBRIS TRACKING SYSTEM

1. Data Collection Section: Mention About the Data that you Collected to Design the Database

A Space Debris Tracking System Database based on the given information can be designed to store and manage data related to satellites, debris objects, and tracking information. Here's an overview of the database structure and some additional information:

1. Tables:

a) Satellites:
- satellite_id (Primary Key): Unique identifier for each satellite.
- name: Name of the satellite.
- launch_date: Date when the satellite was launched.
- country: Country of origin of the satellite.
- mission_status: Status of the satellite's mission.
- orbit_type: Type of orbit the satellite is in.

b) Debris:
- debris_id (Primary Key): Unique identifier for each debris object.
- object_name: Name or identifier of the debris object.
- launch_date: Date when the debris object was launched.
- satellite_id (Foreign Key): Reference to the satellite the debris belongs to.
- orbit_type: Type of orbit the debris object is in.

c) TrackingData:
- tracking_id (Primary Key): Unique identifier for each tracking data entry.
- debris_id (Foreign Key): Reference to the debris object being tracked.
- observation_date: Date of the observation.
- observation_time: Time of the observation.
- observation_location: Location where the observation took place.
- observer_name: Name of the observer.

2. Relationships:

- The "Debris" table has a foreign key, "satellite_id," which references the "Satellites" table. This relationship represents that each debris object is associated with a specific satellite.

- The "TrackingData" table has a foreign key, "debris_id," which references the "Debris" table. This relationship represents that each tracking data entry is associated with a specific debris object.

3. Additional Considerations:

- To enhance the database's functionality and usability, you can consider adding more tables and attributes. For example, you might include tables for tracking stations, observers, and additional attributes related to the debris objects or tracking data.

- You can add indexes on commonly used columns to improve query performance.

- Consider implementing appropriate constraints and validations to ensure data integrity.

- Implementing appropriate security measures, such as access controls and encryption, is crucial to protect sensitive data.

- Regular maintenance and data updates are essential to keep the database accurate and up-to-date.

2. Identification of ER components: identify and present the names of entity sets, relationship sets, types of entity sets (justify), types of attributes (justify), type of relationship sets (justify), etc. You may use the following example table to present these details;

ENTITY SET DETAILS:

NAME OF THE COMPONENT	TYPE	SUB-TYPE	JUSTIFICATION
SATELLITES	ENTITY SET	STRONG ENTITY	
DEBRIS	ENTITY SET	STRONG ENTITY	
TRACKING DATA	ENTITY SET	STRONG ENTITY	

ATTRIBUTE DETAILS:

Sattelite:

ATTRIBUTE NAME	TYPE	ENTITY/RELATIONSHIP SET	ADDITIONAL
Satellite_id[PK]	Single valued	Sattelite	Primary Key
name	composite	Sattelite	
Launch_date	Stored	Sattelite	
country	Composite	Sattelite	
Mission_status	Multivalued	Sattelite	
Orbit_type	Multivalued	Sattelite	

Debris

ATTRIBUTE NAME	TYPE	ENTITY/RELATIONSHIP SET	ADDITIONAL
Debris_id[PK]	Single valued	Debris	Primary key
Object_name	Composite	Debris	
Launch_date	Stored	Debris	
Satellite_id[FK]		Debris	Foreign Key
Orbit_type	Mutivalued	Debris	

Tracking Data

ATTRIBUTE NAME	TYPE	ENTITY/RELATIONSHIP SET	ADDITIONAL
Tracking_id[PK]	Single valued	Tracking Data	Primary Key
Debris_id[FK]		Tracking Data	Foreign Key
Observation_date	Stored	Tracking Data	
Observation_time	Simple	Tracking Data	
Observation_location	Mutlivalued	Tracking Data	
Observation_name	Composite	Tracking Data	

3. Draw ER Diagram: Draw neat ER Diagram For your Database

The ER (Entity-Relationship) diagram represents the relationships between entities in a database. Based on the information you provided, we can describe the structure of the ER diagram as follows:

Entities:

1. Satellites
   - Attributes: satellite_id (primary key), name, launch_date, country, mission_status, orbit_type

2. Debris
   - Attributes: debris_id (primary key), object_name, launch_date, satellite_id (foreign key referencing the Satellites table), orbit_type

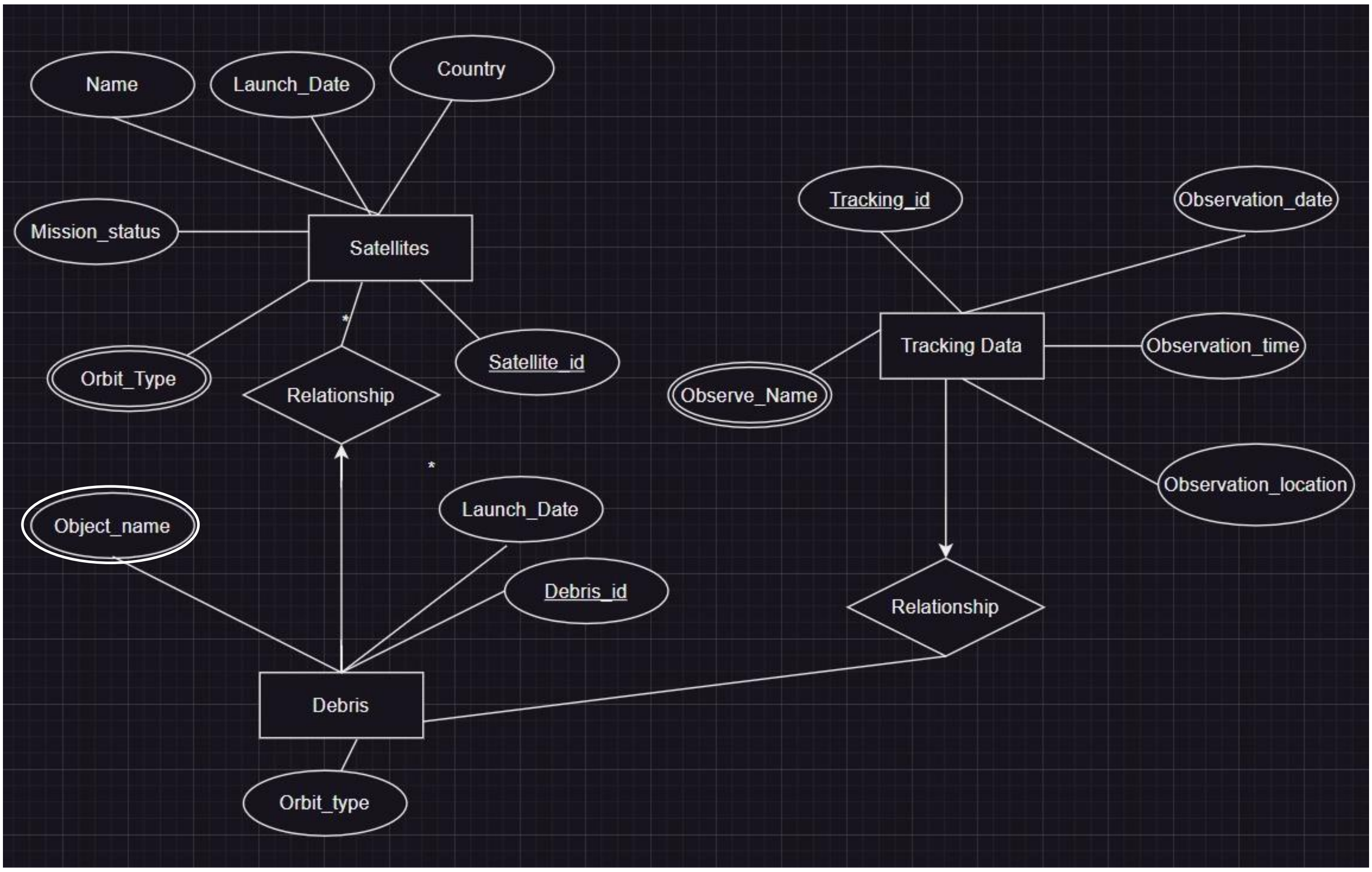
3. TrackingData
   - Attributes: tracking_id (primary key), debris_id (foreign key referencing the Debris table), observation_date, observation_time, observation_location, observer_name

Relationships:

1. Satellites - Debris (One-to-Many):
   - A satellite can have multiple debris objects associated with it.
   - The satellite_id attribute in the Debris table is a foreign key referencing the satellite_id attribute in the Satellites table.

2. Debris - TrackingData (One-to-Many):
   - A debris object can have multiple tracking data entries associated with it.
   - The debris_id attribute in the TrackingData table is a foreign key referencing the debris_id attribute in the Debris table.

Note: The primary key of a table is denoted by [PK], and a foreign key is denoted by [FK]. The [PK] and [FK] annotations indicate the primary key and foreign key constraints in the database schema.



4) Reduce to schema: Convert the ER diagram into set of conceptual schemas by applying the rules. Please justify wherever required. Justification for every database components must be given at least once.

Satellites (satellite_id [PK], name, launch_date, country, mission_status, orbit_type)

Debris (debris_id [PK], object_name, launch_date, satellite_id [FK], orbit_type)

TrackingData (tracking_id [PK], debris_id [FK], observation_date, observation_time, observation_location, observer_name)

5. Normal form definitions – 1NF, 2NF, 3NF, and BCNF.

Database Normalisation

- “Database Normalization” is a process or technique to reduce the attribute redundancy and functional dependency within the set of tables present in any database.
- It can be defined as a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.
- The process of decomposing unsatisfactory relations schema that do not meet certain normal form tests are decomposed into smaller relation schema that meet certain normal form tests(Desirable Properties).
- Normal form - Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form.

Types of Normalization:

➔ First Normal Form (1NF)
➔ Second Normal Form (2NF)
➔ Third Normal Form (3NF)
➔ Boyce-Codd Normal Form (BCNF)

First Normal Form (1NF)

- A relation is in the first normal form iff:

  The domain of an attribute must include atomic values, and

  The value of each attribute in a tuple must be a single value from the domain of that attribute.
- Disallows:

  Composite attributes

  Multi-valued attributes
- Every relation is in 1NF by definition - If not, the database is termed as a bad design.

Second Normal Form (2NF)

- Second normal form (2NF) is based on the concept of full functional dependency.

- A f.d $x \rightarrow y$ is a Full Functional Depedency(FFD), if removal of any attribute from x means that the dependency does not hold any more. if $A \in x$ then $\{x - A\} \nrightarrow y$
- A relational schema R is in 2NF if each attribute A in R satisfies one of the following criteria:
  - ➔ The table must be in the first normal form.
  - ➔ No non-prime attribute is partially dependent on any key of R.
- In other words, no non-prime attribute (not a part of any candidate key) is dependent on a proper subset of any candidate key.
- 2NF tries to reduce redundant information.

Third Normal Form (3NF)

- Third normal form (3NF) is based on the concept of transitive dependency.
- According to Codd's original definition, a relation schema R is in 3NF if it satisfies the following:
  - ➔ It has to be in a 2NF.
  - ➔ There should be no transitivity dependency for non-prime attributes.

(or)

- A relation schema R is in 3NF if, whenever a nontrivial functional dependency $x \rightarrow y$ holds in R, then either
  - ➔ x is a superkey of R or
  - ➔ y is a prime attribute of R.

Boyce-Codd Normal Form (BCNF)

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every functional dependency $X \rightarrow Y$, X is the super key of the table.
- For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

6. Each of your table (that you created using ER model in phase 1) with some sample data.

Satellites:

satellite_id	Name	launch_date	Country	mission_status	orbit_type
1	Satellite A	2022-01-15	USA	Active	LEO, GEO
2	Satellite B	2023-02-10	India	Active	GEO, LEO
3	Satellite C	2020-12-01	Russia	Inactive	LEO
4	Satellite D	2020-12-01	India	Active	LEO
5	Satellite E	2019-06-17	USA	Inactive	GEO, LEO

Debris:

debris_id	object_name	launch_date	satellite_id (FK)	orbit_type
D1	Debris 1	2022-01-15	1	LEO
D2	Debris 2, Debris 1	2023-02-10	2	GEO
D3	Debris 3	2023-02-10	3	LEO
D4	Debris 4, Debris 5	2020-12-01	4	LEO
D5	Debris 5	2019-06-17	5	GEO

TrackingData:

<u>tracking_id</u>	debis_id (FK)	observation_date	observation_time	observation_location	observer_name
T1	D1	2023-01-01	10:30:00	Space Station	John Doe, Jane Smith
T2	D2	2023-03-03	14:45:00	Ground Station	Jane Smith
T3	D3	2023-03-03	22:10:00	Space Station	Mark Johnson, John Doe
T4	D4	2023-04-20	22:10:00	Ground Station	Emily Davis
T5	D5	2023-05-10	16:20:00	Space Station	Michael Wilson,John Doe

7. Identify and list all functional dependencies in each table.

Satellites = {satellite_id ➔ name, launch_date, country, mission_status, orbit_type;
Name ➔ satellite_id, launch_date, country, mission_status, orbit_type;
Name, launch_date ➔ country;
launch_date, country ➔ mission_status, orbit_type;
Name, orbit_type ➔ launch_date, country, mission_status, satellite_id;
launch_date, mission_status ➔ name, country, mission_status, orbit_type;}

Debris = {debris_id ➔ object_name, launch_date, satellite_id [FK], orbit_type;
satellite_id [FK] ➔ object_name, orbit_type, launch_date;
satellite_id [FK], launch_date ➔ object_name, orbit_type;
object_name, launch_date ➔ orbit_type;
object_name, satellite_id [FK] ➔ orbit_type;
satellite_id [FK], orbit_type ➔ object_name, launch_date;

TrackingData = {tracking_id ➔ debris_id [FK], observation_date, observation_time, observation_location, observer_name;
observation_date, observation_time ➔ observation_location, observer_name;
observation_time, observation_location ➔ observer_name, observation_date;
debris_id [FK], observer_name ➔ observation_date, observation_time, observation_location;}

8. Find the candidate keys for each table.

Satellites

To determine the candidate keys using the functional dependencies approach, we can start by examining the given functional dependencies and identify the attributes that uniquely determine all other attributes. Let's go through the functional dependencies one by one:

satellite_id ➔ name, launch_date, country, mission_status, orbit_type:
This functional dependency indicates that satellite_id uniquely determines all other attributes. Therefore, satellite_id is a candidate key on its own.

Name ➔ satellite_id, launch_date, country, mission_status, orbit_type:
This functional dependency states that the attribute Name uniquely determines all other attributes. Hence, Name can also be considered a candidate key.

Name, launch_date ➔ country:
This functional dependency implies that the combination of Name and launch_date determines the attribute country. However, this combination does not uniquely identify each row, as there may be multiple satellites with the same Name and launch_date. Therefore, it is not a candidate key.

launch_date, country ➔ mission_status, orbit_type:
Similarly, this functional dependency shows that the combination of launch_date and country determines mission_status and orbit_type. However, this combination does not uniquely identify each row due to the possibility of multiple satellites with the same launch_date and country. Therefore, it is not a candidate key.

Name, orbit_type ➔ launch_date, country, mission_status, satellite_id:
This functional dependency indicates that the combination of Name and orbit_type determines all other attributes, including launch_date, country, mission_status, and satellite_id. Hence, the combination of Name and orbit_type can be considered a candidate key.

launch_date, mission_status ➔ name, country, mission_status, orbit_type:
Similar to the previous cases, this functional dependency does not uniquely determine all other attributes, as there may be multiple satellites with the same launch_date and mission_status. Therefore, it is not a candidate key.

Based on the functional dependencies, the candidate keys for the table are:

- *satellite_id*
- *Name, orbit_type*

Debris
To determine the candidate keys using the functional dependencies approach, let's analyze the given functional dependencies and identify the attributes that uniquely determine all other attributes. Here are the functional dependencies provided:

debris_id ➔ object_name, launch_date, satellite_id [FK], orbit_type:
This functional dependency states that debris_id uniquely determines object_name, launch_date, satellite_id, and orbit_type. Hence, debris_id is a candidate key on its own.

satellite_id [FK] ➔ object_name, orbit_type, launch_date:
This functional dependency indicates that the attribute satellite_id uniquely determines object_name, orbit_type, and launch_date. Therefore, satellite_id can be considered a candidate key.

satellite_id [FK], launch_date ➔ object_name, orbit_type:
This functional dependency shows that the combination of satellite_id and launch_date determines object_name and orbit_type. However, this combination does not uniquely identify each row, as there may be multiple records with the same satellite_id and launch_date. Hence, it is not a candidate key.

object_name, launch_date ➔ orbit_type:
This functional dependency implies that the combination of object_name and launch_date uniquely determines orbit_type. Thus, the combination of object_name and launch_date can be considered a candidate key.

object_name, satellite_id [FK] ➔ orbit_type:
This functional dependency indicates that the combination of object_name and satellite_id uniquely determines orbit_type. Hence, the combination of object_name and satellite_id is a candidate key.

satellite_id [FK], orbit_type ➔ object_name, launch_date:
Similarly, this functional dependency shows that the combination of satellite_id and orbit_type determines object_name and launch_date. Therefore, the combination of satellite_id and orbit_type is a candidate key.

Based on the functional dependencies, the candidate keys for the table are:

- *debris_id*
- *satellite_id*

- *object_name, launch_date*
- *object_name, satellite_id*
- *satellite_id, orbit_type*

TrackingData

Let's analyze the given functional dependencies and determine the candidate keys using the functional dependencies approach. Here are the functional dependencies provided:

tracking_id ➔ debris_id [FK], observation_date, observation_time, observation_location, observer_name:
This functional dependency indicates that tracking_id uniquely determines all other attributes. Therefore, tracking_id is a candidate key on its own.

observation_date, observation_time ➔ observation_location, observer_name:
This functional dependency shows that the combination of observation_date and observation_time uniquely determines observation_location and observer_name. Hence, the combination of observation_date and observation_time can be considered a candidate key.

observation_time, observation_location ➔ observer_name, observation_date:
Similarly, this functional dependency implies that the combination of observation_time and observation_location uniquely determines observer_name and observation_date. Thus, the combination of observation_time and observation_location can be considered a candidate key.

debris_id [FK], observer_name ➔ observation_date, observation_time, observation_location:
This functional dependency indicates that the combination of debris_id and observer_name uniquely determines observation_date, observation_time, and observation_location. Hence, the combination of debris_id and observer_name is a candidate key.

Based on the functional dependencies, the candidate keys for the table are:

- *tracking_id*
- *observation_date, observation_time*
- *observation_time, observation_location*
- *debris_id, observer_name*

9. Normalize each table upto BCNF. Show the detailed steps for at least two tables. All the other tables, you can directly show the normalized schemas.

First Normal Form (1NF):

Satellite Table:

satellite_id	Name	launch_date	Country	mission_status	orbit_type
1	Satellite A	2022-01-15	USA	Active	LEO, GEO
2	Satellite B	2023-02-10	India	Active	GEO, LEO
3	Satellite C	2020-12-01	Russia	Inactive	LEO
4	Satellite D	2020-12-01	India	Active	LEO
5	Satellite E	2019-06-17	USA	Inactive	GEO, LEO

First Normal Form (1NF) form of Satellite Table:

satellite_id	Name	launch_date	Country	mission_status	orbit_type
1	Satellite A	2022-01-15	USA	Active	LEO
1	Satellite A	2022-01-15	USA	Active	GEO
2	Satellite B	2023-02-10	India	Active	GEO
2	Satellite B	2023-02-10	India	Active	LEO
3	Satellite C	2020-12-01	Russia	Inactive	LEO
4	Satellite D	2020-12-01	India	Active	LEO
5	Satellite E	2019-06-17	USA	Inactive	GEO
5	Satellite E	2019-06-17	USA	Inactive	LEO

Here we have created a separate row for the Satellite Table for the multi-valued attribute orbit_type. This is how we will convert the Satellite Table into First Normal Form (1NF).

Debris Table:

debris_id	object_name	launch_date	satellite_id (FK)	orbit_type
D1	Debris 1	2022-01-15	1	LEO
D2	Debris 2, Debris 1	2023-02-10	2	GEO
D3	Debris 3	2023-02-10	3	LEO
D4	Debris 4, Debris 5	2020-12-01	4	LEO
D5	Debris 5	2019-06-17	5	GEO

First Normal Form (1NF) form of Debris Table:

debris_id	object_name	launch_date	satellite_id (FK)	orbit_type
D1	Debris 1	2022-01-15	1	LEO
D2	Debris 1	2023-02-10	2	GEO
D2	Debris 2	2023-02-10	2	GEO
D3	Debris 3	2023-02-10	3	LEO
D4	Debris 4	2020-12-01	4	LEO
D4	Debris 5	2020-12-01	4	LEO
D5	Debris 5	2019-06-17	5	GEO

Here we have created a separate row for the Debris Table for the multi-valued attribute object_name.
This is how we will convert the Debris Table into First Normal Form (1NF).

TrackingData Table:

tracking_id	**debis_id (FK)**	**observation_date**	**observation_time**	**observation_location**	**observer_name**
T1	D1	2023-01-01	10:30:00	Space Station	John Doe, Jane Smith
T2	D2	2023-03-03	14:45:00	Ground Station	Jane Smith
T3	D3	2023-03-03	22:10:00	Space Station	Mark Johnson, John Doe
T4	D4	2023-04-20	22:10:00	Ground Station	Emily Davis
T5	D5	2023-05-10	16:20:00	Space Station	Michael Wilson, John Doe

First Normal Form (1NF) form of TrackingData Table:

tracking_id	**debis_id (FK)**	**observation_date**	**observation_time**	**observation_location**	**observer_name**
T1	D1	2023-01-01	10:30:00	Space Station	John Doe
T1	D1	2023-01-01	10:30:00	Space Station	Jane Smith
T2	D2	2023-03-03	14:45:00	Ground Station	Jane Smith
T3	D3	2023-03-03	22:10:00	Space Station	Mark Johnson,
T3	D3	2023-03-03	22:10:00	Space Station	John Doe
T4	D4	2023-04-20	22:10:00	Ground Station	Emily Davis
T5	D5	2023-05-10	16:20:00	Space Station	Michael Wilson
T5	D5	2023-05-10	16:20:00	Space Station	John Doe

Here we have created a separate row for the TrackingData Table for the multi-valued attribute observer_name.
This is how we will convert the TrackingData Table into First Normal Form (1NF).

Second Normal Form (2NF):

Satellite Table:

satellite_id	Name	launch_date	Country	mission_status	orbit_type
1	Satellite A	2022-01-15	USA	Active	LEO, GEO
2	Satellite B	2023-02-10	India	Active	GEO, LEO
3	Satellite C	2020-12-01	Russia	Inactive	LEO
4	Satellite D	2020-12-01	India	Active	LEO
5	Satellite E	2019-06-17	USA	Inactive	GEO, LEO

Second Normal Form (2NF) form of Satellite Table:

To convert the given table into 2NF (Second Normal Form), we need to ensure that each non-key attribute is fully dependent on the entire primary key. We will start by identifying the functional dependencies:

Primary Key: satellite_id
Attributes: Name, launch_date, Country, mission_status, orbit_type

Based on the given data, we can determine the functional dependencies as follows:

satellite_id ➔ Name, launch_date, Country, mission_status, orbit_type
Since there are no partial dependencies, the table is already in 2NF.

However, we can still separate the attributes related to mission_status and orbit_type into their own table to improve normalization:

Table 1: Satellites
Attributes: satellite_id (PK), Name, launch_date, Country

Table 2: Satellite_Details
Attributes: satellite_id (FK), mission_status, orbit_type

Here is the updated tabular representation:

	Name	launch_date	Country
1	Satellite A	2022-01-15	USA
2	Satellite B	2023-02-10	India
3	Satellite C	2020-12-01	Russia
4	Satellite D	2020-12-01	India
5	Satellite E	2019-06-17	USA

satellite_id	**mission_status**	**orbit_type**
1	Active	LEO, GEO
2	Active	GEO, LEO
3	Inactive	LEO
4	Active	LEO
5	Inactive	GEO, LEO

Debris Table:

debris_id	object_name	launch_date	satellite_id (FK)	orbit_type
D1	Debris 1	2022-01-15	1	LEO
D2	Debris 2, Debris 1	2023-02-10	2	GEO
D3	Debris 3	2023-02-10	3	LEO
D4	Debris 4, Debris 5	2020-12-01	4	LEO
D5	Debris 5	2019-06-17	5	GEO

Second Normal Form (2NF) form of Debris Table:

To convert the given table into 2NF (Second Normal Form), we need to ensure that each non-key attribute is fully dependent on the entire primary key. We will start by identifying the functional dependencies:

Primary Key: debris_id
Attributes: object_name, launch_date, satellite_id (FK), orbit_type

Based on the given data, we can determine the functional dependencies as follows:

debris_id ➔ object_name, launch_date, satellite_id (FK), orbit_type
Since there are no partial dependencies, the table is already in 2NF.

However, we can still separate the attributes related to satellite information into their own table to improve normalization:

Table 1: Debris
Attributes: debris_id (PK), object_name, launch_date

Table 2: Debris_Details
Attributes: debris_id (FK), satellite_id (FK), orbit_type

Here is the updated tabular representation:

debris_id	**object_name**	**launch_date**
D1	Debris 1	2022-01-15
D2	Debris 2, Debris 1	2023-02-10
D3	Debris 3	2023-02-10
D4	Debris 4, Debris 5	2020-12-01
D5	Debris 5	2019-06-17

debris_id	satellite_id (FK)	orbit_type
D1	1	LEO
D2	2	GEO
D3	3	LEO
D4	4	LEO
D5	5	GEO

TrackingData Table:

tracking_id	debis_id (FK)	observation_date	observation_time	observation_location	observer_name
T1	D1	2023-01-01	10:30:00	Space Station	John Doe, Jane Smith
T2	D2	2023-03-03	14:45:00	Ground Station	Jane Smith
T3	D3	2023-03-03	22:10:00	Space Station	Mark Johnson, John Doe
T4	D4	2023-04-20	22:10:00	Ground Station	Emily Davis
T5	D5	2023-05-10	16:20:00	Space Station	Michael Wilson, John Doe

Second Normal Form (2NF) form of TrackingData Table:

To convert the given table into 2NF (Second Normal Form), we need to ensure that each non-key attribute is fully dependent on the entire primary key. We will start by identifying the functional dependencies:

Primary Key: tracking_id
Attributes: debris_id (FK), observation_date, observation_time, observation_location, observer_name

Based on the given data, we can determine the functional dependencies as follows:

tracking_id ➔ debris_id (FK), observation_date, observation_time, observation_location, observer_name
Since there are no partial dependencies, the table is already in 2NF.

However, we can further normalize the table by separating the observer_name attribute into a separate table, as it appears to contain multiple values.

Table 1: TrackingData
Attributes: tracking_id (PK), debris_id (FK), observation_date, observation_time, observation_location

Table 2: Observers
Attributes: tracking_id (FK), observer_name

Here is the updated tabular representation:

<u>tracking_id</u>	**debis_id (FK)**	**observation_date**	**observation_time**	**observation_location**
T1	D1	2023-01-01	10:30:00	Space Station
T2	D2	2023-03-03	14:45:00	Ground Station
T3	D3	2023-03-03	22:10:00	Space Station
T4	D4	2023-04-20	22:10:00	Ground Station
T5	D5	2023-05-10	16:20:00	Space Station

<u>tracking_id</u>	**observer_name**
T1	John Doe, Jane Smith
T2	Jane Smith
T3	Mark Johnson, John Doe
T4	Emily Davis
T5	Michael Wilson, John Doe

Third Normal Form (3NF):

A relation schema R is in 3NF
if it satisfies the following:

➔ It must be in a 2NF.

➔ There should be no transitivity dependency for non-prime attributes.

Satellite Table:

Satellite Table is already in 2NF and there is no transitivity dependency. So 3NF is same as 2NF.

satellite_id	Name	launch_date	Country
1	Satellite A	2022-01-15	USA
2	Satellite B	2023-02-10	India
3	Satellite C	2020-12-01	Russia
4	Satellite D	2020-12-01	India
5	Satellite E	2019-06-17	USA

satellite_id	mission_status	orbit_type
1	Active	LEO, GEO
2	Active	GEO, LEO
3	Inactive	LEO
4	Active	LEO
5	Inactive	GEO, LEO

Debris Table:

Both tables are already in 3NF, as there are no dependencies that violate the normalization principles. Each table represents a separate entity, and the attributes within each table are dependent on the primary key.

debris_id	object_name	launch_date
D1	Debris 1	2022-01-15
D2	Debris 2, Debris 1	2023-02-10
D3	Debris 3	2023-02-10
D4	Debris 4, Debris 5	2020-12-01
D5	Debris 5	2019-06-17

debris_id	satellite_id (FK)	orbit_type
D1	1	LEO
D2	2	GEO
D3	3	LEO
D4	4	LEO
D5	5	GEO

TrackingData Table:

TrackingData Table is already in 2NF and there is no transitivity dependency. So 3NF is same as 2NF.

tracking_id	debis_id (FK)	observation_date	observation_time	observation_location
T1	D1	2023-01-01	10:30:00	Space Station
T2	D2	2023-03-03	14:45:00	Ground Station
T3	D3	2023-03-03	22:10:00	Space Station
T4	D4	2023-04-20	22:10:00	Ground Station
T5	D5	2023-05-10	16:20:00	Space Station

tracking_id	observer_name
T1	John Doe, Jane Smith
T2	Jane Smith
T3	Mark Johnson, John Doe
T4	Emily Davis
T5	Michael Wilson, John Doe

BCNF:

Satellite Table:

The table already satisfies the requirements of BCNF since there are no overlapping or redundant dependencies and it is also in 3NF.

satellite_id	Name	launch_date	Country
1	Satellite A	2022-01-15	USA
2	Satellite B	2023-02-10	India
3	Satellite C	2020-12-01	Russia
4	Satellite D	2020-12-01	India
5	Satellite E	2019-06-17	USA

satellite_id	mission_status	orbit_type
1	Active	LEO, GEO
2	Active	GEO, LEO
3	Inactive	LEO
4	Active	LEO
5	Inactive	GEO, LEO

Debris Table:

The table already satisfies the requirements of BCNF since there are no overlapping or redundant dependencies and it is also in 3NF.

debris_id	**object_name**	**launch_date**
D1	Debris 1	2022-01-15
D2	Debris 2, Debris 1	2023-02-10
D3	Debris 3	2023-02-10
D4	Debris 4, Debris 5	2020-12-01
D5	Debris 5	2019-06-17

debris_id	**satellite_id (FK)**	**orbit_type**
D1	1	LEO
D2	2	GEO
D3	3	LEO
D4	4	LEO
D5	5	GEO

TrackingData Table:

The table already satisfies the requirements of BCNF since there are no overlapping or redundant dependencies and it is also in 3NF.

tracking_id	**debis_id (FK)**	**observation_date**	**observation_time**	**observation_location**
T1	D1	2023-01-01	10:30:00	Space Station
T2	D2	2023-03-03	14:45:00	Ground Station
T3	D3	2023-03-03	22:10:00	Space Station
T4	D4	2023-04-20	22:10:00	Ground Station
T5	D5	2023-05-10	16:20:00	Space Station

tracking_id	**observer_name**
T1	John Doe, Jane Smith
T2	Jane Smith
T3	Mark Johnson, John Doe
T4	Emily Davis
T5	Michael Wilson, John Doe

10.List the schemas for all the normalized tables as final result.

Normalization:

- Satellites (satellite_id [PK], name, launch_date, country, mission_status, orbit_type)
- Debris (debris_id [PK], object_name, launch_date, satellite_id [FK], orbit_type)
- TrackingData (tracking_id [PK], debris_id [FK], observation_date, observation_time, observation_location, observer_name)

BCNF Normalization:

Satellites

- Satellites (satellite_id [PK], name, launch_date, country)
- Satellites (satellite_id [PK], mission_status, orbit_type)

Debris

- Debris (debris_id [PK], object_name, launch_date
- Debris (debris_id [PK], satellite_id [FK], orbit_type)

TrackingData

- TrackingData (tracking_id [PK], debris_id [FK], observation_date, observation_time, observation_location)
- TrackingData (tracking_id [PK], observer_name)

Database schema:

1. Satellites table (satellite_id (primary key), name, launch_date, country, mission_status, orbit_type)

2. Debris table(debris_id (primary key), object_name, launch_date, satellite_id (foreign key referencing the Satellites table), orbit_type)

3. TrackingDatatable (tracking_id (primary key), debris_id (foreign key referencing the Debris table), observation_date, observation_time, observation_location, observer_name)

CREATE TABLE:

Satellites

```
SQL> DESC Satellites;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SATELLITE_ID                              NOT NULL VARCHAR2(10)
 NAME                                      NOT NULL VARCHAR2(20)
 LAUNCH_DATE                               NOT NULL DATE
 COUNTRY                                   NOT NULL VARCHAR2(20)
 MISSION_STATUS                            NOT NULL VARCHAR2(20)
 ORBIT_TYPE                                         VARCHAR2(20)
```

Debris

```
SQL> DESC Debris;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEBRIS_ID                                 NOT NULL VARCHAR2(10)
 OBJECT_NAME                               NOT NULL VARCHAR2(20)
 LAUNCH_DATE                               NOT NULL DATE
 SATELLITE_ID                                       VARCHAR2(10)
 ORBIT_TYPE                                         VARCHAR2(20)
```

TrackingData

```
SQL> DESC TrackingData;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 TRACKING_ID                               NOT NULL VARCHAR2(10)
 DEBRIS_ID                                          VARCHAR2(10)
 OBSERVATION_DATE                          NOT NULL DATE
 OBSERVATION_TIME                          NOT NULL VARCHAR2(10)
 OBSERVATION_LOCATION                      NOT NULL VARCHAR2(20)
 OBSERVER_NAME                                      VARCHAR2(20)
```

CREATE TABLE statements:

CREATE TABLE Satellites (

satellite_id VARCHAR(10) PRIMARY KEY,

name VARCHAR(20) NOT NULL,

launch_date DATE NOT NULL,

country VARCHAR(20) NOT NULL,

mission_status VARCHAR(20) NOT NULL,

orbit_type VARCHAR(20),

CONSTRAINT satellites_unique_name UNIQUE (name)

);

```
SQL> CREATE TABLE Satellites (
  2         satellite_id VARCHAR(10) PRIMARY KEY,
  3         name VARCHAR(20) NOT NULL,
  4         launch_date DATE NOT NULL,
  5         country VARCHAR(20) NOT NULL,
  6         mission_status VARCHAR(20) NOT NULL,
  7         orbit_type VARCHAR(20),
  8         CONSTRAINT satellites_unique_name UNIQUE (name)
  9       );

Table created.
```

CREATE TABLE Debris (

debris_id VARCHAR(10) PRIMARY KEY,

object_name VARCHAR(20) NOT NULL,

launch_date DATE NOT NULL,

satellite_id REFERENCES Satellites(satellite_id),

orbit_type VARCHAR(20),

CONSTRAINT debris_unique_name UNIQUE (object_name)

);

```
SQL> CREATE TABLE Debris (
  2    debris_id VARCHAR(10) PRIMARY KEY,
  3    object_name VARCHAR(20) NOT NULL,
  4    launch_date DATE NOT NULL,
  5    satellite_id REFERENCES Satellites(satellite_id),
  6    orbit_type VARCHAR(20),
  7    CONSTRAINT debris_unique_name UNIQUE (object_name)
  8  );

Table created.
```

CREATE TABLE TrackingData (

tracking_id VARCHAR(10) PRIMARY KEY,

debris_id REFERENCES Debris(debris_id),

observation_date DATE NOT NULL,

observation_time VARCHAR(10) NOT NULL,

observation_location VARCHAR(20) NOT NULL,

observer_name VARCHAR(20));

```
SQL> CREATE TABLE TrackingData (
  2         tracking_id VARCHAR(10) PRIMARY KEY,
  3         debris_id REFERENCES Debris(debris_id),
  4         observation_date DATE NOT NULL,
  5         observation_time VARCHAR(10) NOT NULL,
  6         observation_location VARCHAR(20) NOT NULL,
  7         observer_name VARCHAR(20));

Table created.
```

INSERT statement:

Satellites:

INSERT INTO Satellites VALUES ('S1','Sat A','15-JAN-2022','INDIA','Active','LEO');

INSERT INTO Satellites VALUES ('S2','Sat B','10-FEB-2023','CHINA','Active','GEO');

INSERT INTO Satellites VALUES ('S3','Sat C','23-JUL-2021','UK','InActive','LEO');

INSERT INTO Satellites VALUES ('S4','Sat D','19-OCT-2019','RUSSIA','Active','LEO');

INSERT INTO Satellites VALUES ('S5','Sat E','13-DEC-2023','US','InActive','GEO');

```
SQL> INSERT INTO Satellites  VALUES ('S1','Sat A','15-JAN-2022','INDIA','Active','LEO');

1 row created.

SQL> INSERT INTO Satellites  VALUES ('S2','Sat B','10-FEB-2023','CHINA','Active','GEO');

1 row created.

SQL> INSERT INTO Satellites VALUES ('S3','Sat C','23-JUL-2021','UK','InActive','LEO');

1 row created.

SQL> INSERT INTO Satellites VALUES ('S4','Sat D','19-OCT-2019','RUSSIA','Active','LEO');

1 row created.

SQL> INSERT INTO Satellites VALUES ('S5','Sat E','13-DEC-2023','US','InActive','GEO');

1 row created.
```

```
SATELLITE_ NAME                 LAUNCH_DA COUNTRY
---------- -------------------- --------- --------------------
MISSION_STATUS       ORBIT_TYPE
-------------------- --------------------
S1         Sat A                15-JAN-22 INDIA
Active               LEO

S2         Sat B                10-FEB-23 CHINA
Active               GEO

S3         Sat C                23-JUL-21 UK
InActive             LEO


SATELLITE_ NAME                 LAUNCH_DA COUNTRY
---------- -------------------- --------- --------------------
MISSION_STATUS       ORBIT_TYPE
-------------------- --------------------
S4         Sat D                19-OCT-19 RUSSIA
Active               LEO

S5         Sat E                13-DEC-23 US
InActive             GEO
```

DEBRIS

INSERT INTO DEBRIS VALUES ('D1','Debris 1','15-JAN-22','S1','LEO');

INSERT INTO DEBRIS VALUES ('D2','Debris 2','10-FEB-23','S2','GEO');

INSERT INTO DEBRIS VALUES ('D3','Debris 3','23-JUL-21','S3','LEO');

INSERT INTO DEBRIS VALUES ('D4','Debris 4','19-OCT-19','S4','LEO');

INSERT INTO DEBRIS VALUES ('D5','Debris 5','13-DEC-23','S5','GEO');

```
SQL> INSERT INTO DEBRIS VALUES ('D1','Debris 1','15-JAN-22','S1','LEO');

1 row created.

SQL> INSERT INTO DEBRIS VALUES ('D2','Debris 2','10-FEB-23','S2','GEO');

1 row created.

SQL> INSERT INTO DEBRIS VALUES ('D3','Debris 3','23-JUL-21','S3','LEO');

1 row created.

SQL> INSERT INTO DEBRIS VALUES ('D4','Debris 4','19-OCT-19','S4','LEO');

1 row created.

SQL> INSERT INTO DEBRIS VALUES ('D5','Debris 5','13-DEC-23','S5','GEO');

1 row created.

SQL> SELECT * FROM DEBRIS;
```

```
DEBRIS_ID  OBJECT_NAME          LAUNCH_DA SATELLITE_ ORBIT_TYPE
---------- -------------------- --------- ---------- --------------------
D1         Debris 1             15-JAN-22 S1         LEO
D2         Debris 2             10-FEB-23 S2         GEO
D3         Debris 3             23-JUL-21 S3         LEO
D4         Debris 4             19-OCT-19 S4         LEO
D5         Debris 5             13-DEC-23 S5         GEO
```

TRACKINGDATA:

INSERT INTO TRACKINGDATA VALUES ('T1','D1','01-JAN-2023','10:30:00','Space Station','GUNJAN SAXENA');

INSERT INTO TRACKINGDATA VALUES ('T2','D2','15-FEB-2023','14:45:00','Ground Station','PREETI SINGH');

INSERT INTO TRACKINGDATA VALUES ('T3','D3','03-MAR-2023','08:15:00','Space Station','LAXMAN RAVI');

INSERT INTO TRACKINGDATA VALUES ('T4','D4','20-APR-2023','22:10:00','Ground Station','KISHORE RATHORE');

INSERT INTO TRACKINGDATA VALUES ('T5','D5','10-MAY-2023','16:20:00','Space Station','M PRAJITHA');

```
SQL> INSERT INTO TRACKINGDATA VALUES ('T1','D1','01-JAN-2023','10:30:00','Space Station','GUNJAN SAXENA');

1 row created.

SQL> INSERT INTO TRACKINGDATA VALUES ('T2','D2','15-FEB-2023','14:45:00','Ground Station','PREETI SINGH');

1 row created.

SQL> INSERT INTO TRACKINGDATA VALUES ('T3','D3','03-MAR-2023','08:15:00','Space Station','LAXMAN RAVI');

1 row created.

SQL> INSERT INTO TRACKINGDATA VALUES ('T4','D4','20-APR-2023','22:10:00','Ground Station','KISHORE RATHORE');

1 row created.

SQL> INSERT INTO TRACKINGDATA VALUES ('T5','D5','10-MAY-2023','16:20:00','Space Station','M PRAJITHA');

1 row created.
```

```
TRACKING_I DEBRIS_ID  OBSERVATI OBSERVATIO OBSERVATION_LOCATION
---------- ---------- --------- ---------- --------------------
OBSERVER_NAME
--------------------
T1         D1         01-JAN-23 10:30:00   Space Station
GUNJAN SAXENA

T2         D2         15-FEB-23 14:45:00   Ground Station
PREETI SINGH

T3         D3         03-MAR-23 08:15:00   Space Station
LAXMAN RAVI


TRACKING_I DEBRIS_ID  OBSERVATI OBSERVATIO OBSERVATION_LOCATION
---------- ---------- --------- ---------- --------------------
OBSERVER_NAME
--------------------
T4         D4         20-APR-23 22:10:00   Ground Station
KISHORE RATHORE

T5         D5         10-MAY-23 16:20:00   Space Station
M PRAJITHA
```

Queries:

ORDER BY:

1) Retrieve the object names, launch dates and satellite id from the DEBRIS table, ordered by the launch date in descending order.

SELECT OBJECT_NAME, LAUNCH_DATE

FROM DEBRIS

ORDER BY LAUNCH_DATE DESC;

```
SQL> SELECT OBJECT_NAME, LAUNCH_DATE, SATELLITE_ID
  2  FROM DEBRIS
  3  ORDER BY LAUNCH_DATE DESC;

OBJECT_NAME          LAUNCH_DA SATELLITE_
-------------------- --------- ----------
Debris 5             13-DEC-23 S5
Debris 2             10-FEB-23 S2
Debris 1             15-JAN-22 S1
Debris 3             23-JUL-21 S3
Debris 4             19-OCT-19 S4

SQL>
```

2) Retrieve the satellite names , launch dates and country from the SATELLITES table, ordered by the launch date in ascending order.

SELECT NAME, LAUNCH_DATE, COUNTRY

FROM Satellites

ORDER BY LAUNCH_DATE ASC;

```
SQL> SELECT NAME, LAUNCH_DATE, COUNTRY
  2  FROM Satellites
  3  ORDER BY LAUNCH_DATE ASC;

NAME                 LAUNCH_DA COUNTRY
-------------------- --------- --------------------
Sat D                19-OCT-19 RUSSIA
Sat C                23-JUL-21 UK
Sat A                15-JAN-22 INDIA
Sat B                10-FEB-23 CHINA
Sat E                13-DEC-23 US

SQL>
```

Aggregate function:

3) Retrieve the count of inactive satellites from the SATELLITES table.

SELECT COUNT(*)

FROM SATELLITES

WHERE MISSION_STATUS = 'InActive';

```
SQL> SELECT COUNT(*)
  2  FROM SATELLITES
  3  WHERE MISSION_STATUS = 'InActive';

  COUNT(*)
----------
         2

SQL>
```

4) Retrieve the maximum launch date from the DEBRIS table.

SELECT MAX(LAUNCH_DATE)

FROM DEBRIS;

```
SQL> SELECT MAX(LAUNCH_DATE)
  2  FROM DEBRIS;

MAX(LAUNC
---------
13-DEC-23
```

Aggregate functions with GROUP BY

5) Retrieve the debris id and the count of object name for each debris id from the DEBRIS table.

SELECT DEBRIS_ID, COUNT(*) AS OBJECT_NAME

FROM DEBRIS

GROUP BY DEBRIS_ID ;

```
SQL> SELECT DEBRIS_ID, COUNT(*) AS OBJECT_NAME
  2  FROM DEBRIS
  3  GROUP BY DEBRIS_ID ;

DEBRIS_ID  OBJECT_NAME
---------- -----------
D1                   1
D2                   1
D3                   1
D4                   1
D5                   1

SQL>
```

6) Retrieve the name and the count of satellite count for each name from the SATELLITES table.

SELECT NAME, COUNT(*) AS SATELLITE_COUNT

FROM SATELLITES

GROUP BY NAME;

```
SQL> SELECT  NAME, COUNT(*) AS SATELLITE_COUNT
  2  FROM SATELLITES
  3  GROUP BY  NAME;

NAME                 SATELLITE_COUNT
-------------------- ---------------
Sat A                              1
Sat B                              1
Sat C                              1
Sat D                              1
Sat E                              1

SQL>
```

7) Retrieve the orbit type and the maximum launch date for each orbit type from the SATELLITE stable.

SELECT ORBIT_TYPE, MAX(LAUNCH_DATE) AS MAX_LAUNCH_DATE

FROM SATELLITES

GROUP BY ORBIT_TYPE;

```
SQL> SELECT ORBIT_TYPE, MAX(LAUNCH_DATE) AS MAX_LAUNCH_DATE
  2  FROM SATELLITES
  3  GROUP BY ORBIT_TYPE;

ORBIT_TYPE           MAX_LAUNC
-------------------- ---------
LEO                  15-JAN-22
GEO                  13-DEC-23

SQL>
```

Aggregate functions + GROUP BY + HAVING:

8) Retrieve the name and the count of satellites for name with more than equal to 1 satellite from the SATELLITES table.

SELECT NAME, COUNT(*) AS SATELLITE_COUNT

FROM SATELLITES

GROUP BY NAME

HAVING COUNT(*) >= 1;

```
SQL> SELECT NAME, COUNT(*) AS SATELLITE_COUNT
  2  FROM SATELLITES
  3  GROUP BY NAME
  4  HAVING COUNT(*) >= 1;

NAME                 SATELLITE_COUNT
-------------------- ---------------
Sat A                              1
Sat B                              1
Sat C                              1
Sat D                              1
Sat E                              1
```

9) Retrieve the name and the maximum launch date for name with more than equal to 1 satellites from the SATELLITES table.

SELECT NAME, MAX(LAUNCH_DATE) AS MAX_LAUNCH_DATE

FROM SATELLITES

GROUP BY NAME

HAVING COUNT(*) >= 1;

```
SQL> SELECT NAME, MAX(LAUNCH_DATE) AS MAX_LAUNCH_DATE
  2  FROM SATELLITES
  3  GROUP BY NAME
  4  HAVING COUNT(*) >= 1;

NAME                 MAX_LAUNC
-------------------- ---------
Sat C                23-JUL-21
Sat B                10-FEB-23
Sat E                13-DEC-23
Sat A                15-JAN-22
Sat D                19-OCT-19

SQL>
```

10) Retrieve the orbit type and the minimum launch date for orbit types with at least 3 satellites from the SATELLITES table.

SELECT ORBIT_TYPE, MIN(LAUNCH_DATE) AS MIN_LAUNCH_DATE

FROM SATELLITES

GROUP BY ORBIT_TYPE

HAVING COUNT(*) <= 3;

```
SQL> SELECT ORBIT_TYPE, MIN(LAUNCH_DATE) AS MIN_LAUNCH_DATE
  2  FROM SATELLITES
  3  GROUP BY ORBIT_TYPE
  4  HAVING COUNT(*) <= 3;

ORBIT_TYPE           MIN_LAUNC
-------------------- ---------
LEO                  19-OCT-19
GEO                  10-FEB-23

SQL>
```

Simple JOIN

11) Retrieve the debris ID, object name,orbit type,tracking id and observation location from the DEBRIS and TRACKINGDATA tables.

SELECT D.DEBRIS_ID, D.OBJECT_NAME,ORBIT_TYPE,T.TRACKING_ID,T.OBSERVATION_LOCATION

FROM DEBRIS D

JOIN TRACKINGDATA T ON D.DEBRIS_ID = T.DEBRIS_ID;

```
SQL> SELECT D.DEBRIS_ID, D.OBJECT_NAME,ORBIT_TYPE,T.TRACKING_ID,T.OBSERVATION_LOCATION
  2  FROM DEBRIS D
  3  JOIN TRACKINGDATA T ON D.DEBRIS_ID = T.DEBRIS_ID;

DEBRIS_ID  OBJECT_NAME          ORBIT_TYPE           TRACKING_I
---------- -------------------- -------------------- ----------
OBSERVATION_LOCATION
--------------------
D1         Debris 1             LEO                  T1
Space Station

D2         Debris 2             GEO                  T2
Ground Station

D3         Debris 3             LEO                  T3
Space Station


DEBRIS_ID  OBJECT_NAME          ORBIT_TYPE           TRACKING_I
---------- -------------------- -------------------- ----------
OBSERVATION_LOCATION
--------------------
D4         Debris 4             LEO                  T4
Ground Station

D5         Debris 5             GEO                  T5
Space Station
```

12) Retrieve the satellite name, launch date,country,observer name and observation location from the SATELLITES and TRACKINGDATA tables.

SELECT S.NAME, S.LAUNCH_DATE,COUNTRY, T.OBSERVER_NAME, T.OBSERVATION_LOCATION

FROM SATELLITES S

JOIN TRACKINGDATA T ON S.LAUNCH_DATE = T.OBSERVATION_DATE;

```
SQL> SELECT S.NAME, S.LAUNCH_DATE,COUNTRY, T.OBSERVER_NAME, T.OBSERVATION_L
  2  FROM SATELLITES S
  3  JOIN TRACKINGDATA T ON S.LAUNCH_DATE = T.OBSERVATION_DATE;

no rows selected
```

13) Retrieve the debris ID, object name, and tracking id from the DEBRIS and TRACKINGDATA tables.

SELECT D.DEBRIS_ID, D.OBJECT_NAME,T.TRACKING_ID

FROM DEBRIS D

JOIN TRACKINGDATA T ON D.DEBRIS_ID = T.DEBRIS_ID;

```
SQL> SELECT D.DEBRIS_ID, D.OBJECT_NAME,T.TRACKING_ID
  2  FROM DEBRIS D
  3  JOIN TRACKINGDATA T ON D.DEBRIS_ID = T.DEBRIS_ID;

DEBRIS_ID  OBJECT_NAME          TRACKING_I
---------- -------------------- ----------
D1         Debris 1             T1
D2         Debris 2             T2
D3         Debris 3             T3
D4         Debris 4             T4
D5         Debris 5             T5

SQL>
```

14) Retrieve the debris ID and tracking id from the DEBRIS and TRACKINGDATA tables.

SELECT D.DEBRIS_ID,T.TRACKING_ID

FROM DEBRIS D

JOIN TRACKINGDATA T ON D.DEBRIS_ID = T.DEBRIS_ID;

```
SQL> SELECT D.DEBRIS_ID,T.TRACKING_ID
  2  FROM DEBRIS D
  3  JOIN TRACKINGDATA T ON D.DEBRIS_ID = T.DEBRIS_ID;

DEBRIS_ID  TRACKING_I
---------- ----------
D1         T1
D2         T2
D3         T3
D4         T4
D5         T5

SQL>
```

Joining more than 2 tables:

15) Retrieve the satellite name and observer name by Joining "SATELLITE," "DEBRIS," and "TRACKINGDATA" tables

SELECT S.NAME, T.OBSERVER_NAME

FROM SATELLITES S

JOIN DEBRIS D ON S.SATELLITE_ID = D.SATELLITE_ID

JOIN TRACKINGDATA T ON D.DEBRIS_ID = T.DEBRIS_ID;

```
SQL> SELECT S.NAME, T.OBSERVER_NAME
  2  FROM SATELLITES S
  3  JOIN DEBRIS D ON S.SATELLITE_ID = D.SATELLITE_ID
  4  JOIN TRACKINGDATA T ON D.DEBRIS_ID = T.DEBRIS_ID;

NAME                 OBSERVER_NAME
-------------------- --------------------
Sat A                GUNJAN SAXENA
Sat B                PREETI SINGH
Sat C                LAXMAN RAVI
Sat D                KISHORE RATHORE
Sat E                M PRAJITHA

SQL>
```

16) Retrieve the satellite name, satellite id object name debris id and observer name by Joining "SATELLITE," "DEBRIS," and "TRACKINGDATA" tables

SELECT S.NAME,S.SATELLITE_ID,D.OBJECT_NAME, T.OBSERVER_NAME,T.DEBRIS_ID

FROM SATELLITES S

JOIN DEBRIS D ON S.SATELLITE_ID = D.SATELLITE_ID

JOIN TRACKINGDATA T ON D.DEBRIS_ID = T.DEBRIS_ID;

```
SQL> SELECT S.NAME,S.SATELLITE_ID,D.OBJECT_NAME, T.OBSERVER_NAME,T.DEBRIS_ID
  2  FROM SATELLITES S
  3  JOIN DEBRIS D ON S.SATELLITE_ID = D.SATELLITE_ID
  4  JOIN TRACKINGDATA T ON D.DEBRIS_ID = T.DEBRIS_ID;

NAME                 SATELLITE_ OBJECT_NAME          OBSERVER_NAME
-------------------- ---------- -------------------- --------------------
DEBRIS_ID
----------
Sat A                S1         Debris 1             GUNJAN SAXENA
D1

Sat B                S2         Debris 2             PREETI SINGH
D2

Sat C                S3         Debris 3             LAXMAN RAVI
D3


NAME                 SATELLITE_ OBJECT_NAME          OBSERVER_NAME
-------------------- ---------- -------------------- --------------------
DEBRIS_ID
----------
Sat D                S4         Debris 4             KISHORE RATHORE
D4

Sat E                S5         Debris 5             M PRAJITHA
D5
```

Joining with JOIN keyword, join condition, and special type of join (outer join)

17) Retrieve the debris id,object name,satellites name,satellite id and orbit type

SELECT D.DEBRIS_ID, D.OBJECT_NAME,S.NAME,S.SATELLITE_ID, S.ORBIT_TYPE

FROM DEBRIS D

RIGHT JOIN SATELLITES S ON D.SATELLITE_ID = S.SATELLITE_ID;

```
SQL> SELECT D.DEBRIS_ID, D.OBJECT_NAME,S.NAME,S.SATELLITE_ID, S.ORBIT_TYPE
  2  FROM DEBRIS D
  3  RIGHT JOIN SATELLITES S ON D.SATELLITE_ID = S.SATELLITE_ID;

DEBRIS_ID  OBJECT_NAME          NAME                 SATELLITE_
---------- -------------------- -------------------- ----------
ORBIT_TYPE
--------------------
D1         Debris 1             Sat A                S1
LEO

D2         Debris 2             Sat B                S2
GEO

D3         Debris 3             Sat C                S3
LEO


DEBRIS_ID  OBJECT_NAME          NAME                 SATELLITE_
---------- -------------------- -------------------- ----------
ORBIT_TYPE
--------------------
D4         Debris 4             Sat D                S4
LEO

D5         Debris 5             Sat E                S5
GEO


SQL>
```

18) Retrieve the satellite name and observer name

SELECT S.NAME, T.OBSERVER_NAME

FROM SATELLITES S

JOIN DEBRIS D ON S.SATELLITE_ID = D.SATELLITE_ID

LEFT JOIN TRACKINGDATA T ON D.DEBRIS_ID = T.DEBRIS_ID;

```
SQL> SELECT S.NAME, T.OBSERVER_NAME
  2  FROM SATELLITES S
  3  JOIN DEBRIS D ON S.SATELLITE_ID = D.SATELLITE_ID
  4  LEFT JOIN TRACKINGDATA T ON D.DEBRIS_ID = T.DEBRIS_ID;

NAME                 OBSERVER_NAME
-------------------- --------------------
Sat A                GUNJAN SAXENA
Sat B                PREETI SINGH
Sat C                LAXMAN RAVI
Sat D                KISHORE RATHORE
Sat E                M PRAJITHA

SQL>
```

19) Retrieve debris id and satellites id, including those with no matching satellite in the SATELLITE table (Left outer join).

SELECT D.DEBRIS_ID, S.SATELLITE_ID

FROM DEBRIS D

LEFT JOIN SATELLITES S ON D.SATELLITE_ID = S.SATELLITE_ID;

```
SQL> SELECT D.DEBRIS_ID, S.SATELLITE_ID
  2  FROM DEBRIS D
  3  LEFT JOIN SATELLITES S ON D.SATELLITE_ID = S.SATELLITE_ID;

DEBRIS_ID  SATELLITE_
---------- ----------
D1         S1
D2         S2
D3         S3
D4         S4
D5         S5

SQL>
```

Queries with string patterns in WHERE clause

20) Retrieve satellites with names starting with "Sat"

SELECT

NAME,COUNTRY,LAUNCH_DATE,MISSION_STAT

US FROM SATELLITES

WHERE NAME LIKE 'Sat%';

```
SQL> SELECT NAME,COUNTRY,LAUNCH_DATE,MISSION_STATUS
  2  FROM SATELLITES
  3  WHERE NAME LIKE 'Sat%';

NAME                 COUNTRY              LAUNCH_DA MISSION_STATUS
-------------------- -------------------- --------- --------------------
Sat A                INDIA                15-JAN-22 Active
Sat B                CHINA                10-FEB-23 Active
Sat C                UK                   23-JUL-21 InActive
Sat D                RUSSIA               19-OCT-19 Active
Sat E                US                   13-DEC-23 InActive

SQL>
```

21) Retrieve satellites with names starting with "S" and ending with "C"

SELECT *

FROM SATELLITES

WHERE NAME LIKE 'S%C';

```
SQL> SELECT *
  2  FROM SATELLITES
  3  WHERE NAME LIKE 'S%C';

SATELLITE_ NAME                 LAUNCH_DA COUNTRY
---------- -------------------- --------- --------------------
MISSION_STATUS       ORBIT_TYPE
-------------------- --------------------
S3         Sat C                23-JUL-21 UK
InActive             LEO


SQL>
```

22) Retrieve debris objects with names containing "5"

SELECT *

FROM DEBRIS

WHERE OBJECT_NAME LIKE '%5%';

```
SQL> SELECT *
  2  FROM DEBRIS
  3  WHERE OBJECT_NAME LIKE '%5%';

DEBRIS_ID  OBJECT_NAME          LAUNCH_DA SATELLITE_ ORBIT_TYPE
---------- -------------------- --------- ---------- --------------------
D5         Debris 5             13-DEC-23 S5         GEO

SQL>
```

Functions

23) Retrieve satellites with orbit type names starting with "L":

SELECT *

FROM SATELLITES

WHERE ORBIT_TYPE LIKE 'L%';

```
SQL> SELECT *
  2  FROM SATELLITES
  3  WHERE ORBIT_TYPE LIKE 'L%';

SATELLITE_ NAME                 LAUNCH_DA COUNTRY
---------- -------------------- --------- --------------------
MISSION_STATUS       ORBIT_TYPE
-------------------- --------------------
S1         Sat A                15-JAN-22 INDIA
Active               LEO

S3         Sat C                23-JUL-21 UK
InActive             LEO

S4         Sat D                19-OCT-19 RUSSIA
Active               LEO
```

24) Retrieve debris name and id with launch dates after a 2022.

SELECT DEBRIS_ID,OBJECT_NAME

FROM DEBRIS

WHERE EXTRACT(YEAR FROM LAUNCH_DATE) > 2022;

```
SQL> SELECT DEBRIS_ID,OBJECT_NAME
  2  FROM DEBRIS
  3  WHERE EXTRACT(YEAR FROM LAUNCH_DATE) > 2022;

DEBRIS_ID  OBJECT_NAME
---------- --------------------
D2         Debris 2
D5         Debris 5
```

25) Retrieve tracking data for observations made on a 15-FEB-23

SELECT TRACKING_ID,OBSERVER_NAME

FROM TRACKINGDATA

WHERE TRUNC(OBSERVATION_DATE) = TO_DATE('1-JAN-2023', 'DD-MM-YYYY');

```
SQL> SELECT TRACKING_ID,OBSERVER_NAME
  2  FROM TRACKINGDATA
  3  WHERE TRUNC(OBSERVATION_DATE) = TO_DATE('1-JAN-2023', 'DD-MM-YYYY');

TRACKING_I OBSERVER_NAME
---------- --------------------
T1         GUNJAN SAXENA

SQL>
```

26) Retrieve satellites with orbit type as an integer value greater than 1

SELECT *

FROM SATELLITES

WHERE ORBIT_TYPE > '1';

Subqueries

27) Retrieve satellite name whose mission status is active

SELECT name FROM Satelllites WHERE mission_status=’Active’;

```
SQL> SELECT name FROM Satellites WHERE mission_status = 'Active';

NAME
--------------------
Sat A
Sat B
Sat D
```

28) Retrieve all debris data whose orbit type is leo.

SELECT * FROM Debris WHERE orbit_type='LEO';

```
SQL> SELECT * FROM Debris WHERE orbit_type = 'LEO';

DEBRIS_ID  OBJECT_NAME          LAUNCH_DA SATELLITE_ ORBIT_TYPE
---------- -------------------- --------- ---------- --------------------
D1         Debris 1             15-JAN-22 S1         LEO
D3         Debris 3             23-JUL-21 S3         LEO
D4         Debris 4             19-OCT-19 S4         LEO
```

29) Retrieve all trackingData data whose debris id is 1.

SELECT * FROM TrackingData WHERE debris_id='D1';

```
SQL> SELECT * FROM TrackingData WHERE debris_id='D1';

TRACKING_I DEBRIS_ID  OBSERVATI OBSERVATIO OBSERVATION_LOCATION
---------- ---------- --------- ---------- --------------------
OBSERVER_NAME
--------------------
T1         D1         01-JAN-23 10:30:00   Space Station
GUNJAN SAXENA
```

****