DBMS

Case study: Chandraayan 3 mission
Draw ER Diagram
Convert into tables and set constraints
fill relevant data
Design 10 queries which covers max concept
Construct one procedure and one function
Enable any one trigger for any of DML actions on table

- ***Creating a case study like this requires a significant amount of detail and work, and it's beyond the scope of a single response. However, I can provide you with an outline of the steps you would need to follow to complete this task.***

Case Study: Chandrayaan 3 Mission

Step 1: ER Diagram

1. Identify the entities involved in the Chandrayaan 3 mission. These could include spacecraft, mission personnel, launch sites, instruments, etc.

2. Define the relationships between these entities (one-to-one, one-to-many, or many-to-many).

3. Create an Entity-Relationship Diagram (ERD) that represents the structure of your database.

Step 2: Convert into Tables and Set Constraints

1. For each entity in your ERD, create a corresponding table in your database schema.

2. Define the columns for each table, including data types and constraints (e.g., primary keys, foreign keys, unique constraints, not null constraints).

3. Create relationships between tables using foreign keys to enforce referential integrity.

Step 3: Fill Relevant Data

1. Populate the tables with relevant data. This data would include information about spacecraft, mission personnel, launch dates, instrument details, mission phases, etc.

Step 4: Design 10 Queries

1. Create a variety of SQL queries that cover different concepts such as SELECT, JOIN, GROUP BY, HAVING, and subqueries.

2. Example queries:

- Retrieve a list of all spacecraft used in Chandrayaan 3.
- Find the names of mission personnel involved in the mission.
- Calculate the total cost of the mission.
- List the instruments used in a specific mission phase.
- Find the launch date of Chandrayaan 3.

Step 5: Construct One Procedure and One Function
1. Create a stored procedure that performs a specific task related to the mission data. For example, a procedure that calculates the total cost of a mission phase.
2. Create a user-defined function that takes parameters and returns a specific result. For example, a function that calculates the age of mission personnel.

Step 6: Enable a Trigger
1. Identify a table and a DML (Data Manipulation Language) action (INSERT, UPDATE, DELETE) on which you want to enable a trigger.
2. Creating a case study like this requires a significant amount of detail and work, and it's beyond the scope of a single response. However, I can provide you with an outline of the steps you would need to follow to complete this task.

ANS: -
Step 1: ER Diagram

Creating an Entity-Relationship Diagram (ERD) for the Chandrayaan 3 mission involves identifying entities, their attributes, and defining relationships between them. Below is a simplified representation of the ERD for the mission:

Entities:

1. *Spacecraft:*

Attributes: Spacecraft_ID (Primary Key), Name, Manufacturer, Launch_Date, Status

2. *Mission_Personnel:*

Attributes: Personnel_ID (Primary Key), First_Name, Last_Name, Role, Date_of_Birth

3. *Launch_Site:*

Attributes: Site_ID (Primary Key), Site_Name, Location, Launch_Date

4. *Instruments:*

Attributes: Instrument_ID (Primary Key), Name, Description, Payload_Type

5. *Mission_Phase:*

Attributes: Phase_ID (Primary Key), Phase_Name, Start_Date, End_Date, Description

Relationships:

1. ***One spacecraft can be used in one or more mission phases.***
- ***Relationship: One-to-Many (Spacecraft to Mission_Phase)***
- ***Foreign Key: Spacecraft_ID in Mission_Phase referencing Spacecraft table.***

2. ***One mission personnel can be assigned to multiple mission phases, and a mission phase can have multiple personnel.***
- ***Relationship: Many-to-Many (Mission_Personnel to Mission_Phase)***
- ***Create a junction table to represent this relationship:***
- ***Personnel_Assignment:***
- ❖ ***Attributes: Assignment_ID (Primary Key), Personnel_ID (Foreign Key), Phase_ID (Foreign Key)***

3. ***Each mission phase can be associated with one launch site.***
- ***Relationship: One-to-One (Mission_Phase to Launch_Site)***
- ***Foreign Key: Launch_Site_ID in Mission_Phase referencing Launch_Site table.***

4. ***One spacecraft can have multiple instruments, and an instrument can be used in multiple spacecrafts.***
- ***Relationship: Many-to-Many (Spacecraft to Instruments)***
- ***Create a junction table to represent this relationship:***
- ***Spacecraft Instrument:***
- ❖ ***Attributes: Relationship_ID (Primary Key), Spacecraft_ID (Foreign Key), Instrument_ID (Foreign Key)***

Step 2: Convert into Tables and Set Constraints
Here, I'll provide you with the SQL table definitions based on the entities identified in the ERD, including the appropriate data types and constraints. I'll also establish the relationships between these tables using foreign keys to enforce referential integrity.

- ***Spacecraft Table:***

```
CREATE TABLE Spacecraft (
    Spacecraft_ID INT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Manufacturer VARCHAR(255),
    Launch_Date DATE,
    Status VARCHAR(50)
);
```

- *Mission_Personnel Table:*

```
CREATE TABLE Mission_Personnel (
   Personnel_ID INT PRIMARY KEY,
   First_Name VARCHAR(50) NOT NULL,
   Last_Name VARCHAR(50) NOT NULL,
   Role VARCHAR(100),
   Date_of_Birth DATE
);
```

- *Launch_Site Table:*

```
CREATE TABLE Launch_Site (
   Site_ID INT PRIMARY KEY,
   Site_Name VARCHAR(100) NOT NULL,
   Location VARCHAR(255),
   Launch_Date DATE
);
```

- *Instruments Table:*

```
CREATE TABLE Instruments (
   Instrument_ID INT PRIMARY KEY,
   Name VARCHAR(100) NOT NULL,
   Description TEXT,
   Payload_Type VARCHAR(50)
);
```

- *Mission_Phase Table:*

```
CREATE TABLE Mission_Phase (
   Phase_ID INT PRIMARY KEY,
   Phase_Name VARCHAR(100) NOT NULL,
   Start_Date DATE,
   End_Date DATE,
   Description TEXT,
   Spacecraft_ID INT, -- Foreign key for relationship
   Launch_Site_ID INT -- Foreign key for relationship
);
```

- *Personnel_Assignment Table (Junction Table for Many-to-Many Relationship):*

```
CREATE TABLE Personnel_Assignment (
   Assignment_ID INT PRIMARY KEY,
   Personnel_ID INT, -- Foreign key for relationship
   Phase_ID INT -- Foreign key for relationship
);
```

- *Spacecraft_Instrument Table (Junction Table for Many-to-Many Relationship):*

```
CREATE TABLE Spacecraft_Instrument (
   Relationship_ID INT PRIMARY KEY,
   Spacecraft_ID INT, -- Foreign key for relationship
   Instrument_ID INT -- Foreign key for relationship
);
```

Now, let's set up the foreign key constraints to establish the relationships between these tables:

```
-- Add foreign key constraints
ALTER TABLE Mission_Phase
ADD FOREIGN KEY (Spacecraft_ID) REFERENCES Spacecraft(Spacecraft_ID);

ALTER TABLE Mission_Phase
ADD FOREIGN KEY (Launch_Site_ID) REFERENCES Launch_Site(Site_ID);

ALTER TABLE Personnel_Assignment
ADD FOREIGN KEY (Personnel_ID) REFERENCES Mission_Personnel(Personnel_ID);

ALTER TABLE Personnel_Assignment
ADD FOREIGN KEY (Phase_ID) REFERENCES Mission_Phase(Phase_ID);

ALTER TABLE Spacecraft_Instrument
ADD FOREIGN KEY (Spacecraft_ID) REFERENCES Spacecraft(Spacecraft_ID);

ALTER TABLE Spacecraft_Instrument
ADD FOREIGN KEY (Instrument_ID) REFERENCES Instruments(Instrument_ID);
```

Step 3: Fill Relevant Data

Populating the tables with relevant data is a crucial step in creating a functional database for the Chandrayaan 3 mission. Below, I'll provide sample data for each table to help you get started. Please note that this is just sample data, and in a real-world scenario, you would have access to the mission's actual data.

- *Spacecraft Table:*

```
-- Sample data for Spacecraft
INSERT INTO Spacecraft (Spacecraft_ID, Name, Manufacturer, Launch_Date, Status)
VALUES
  (1, 'Chandrayaan 3', 'ISRO', '2023-01-15', 'Active'),
  (2, 'Lunar Rover 1', 'ISRO', '2023-01-15', 'Active');
```

- *Mission_Personnel Table:*

```
-- Sample data for Mission_Personnel
INSERT INTO Mission_Personnel (Personnel_ID, First_Name, Last_Name, Role, Date_of_Birth)
VALUES
  (1, 'John', 'Smith', 'Mission Director', '1975-05-10'),
  (2, 'Alice', 'Johnson', 'Astronomer', '1980-09-22');
```

- *Launch_Site Table:*

```
-- Sample data for Launch_Site
INSERT INTO Launch_Site (Site_ID, Site_Name, Location, Launch_Date)
VALUES
  (1, 'Satish Dhawan Space Center', 'Sriharikota, India', '2023-01-15');
```

- *Instruments Table:*

```
-- Sample data for Instruments
INSERT INTO Instruments (Instrument_ID, Name, Description, Payload_Type)
VALUES
  (1, 'High-Resolution Camera', 'Captures high-resolution lunar images.', 'Camera'),
  (2, 'Spectrometer', 'Analyzes the composition of lunar soil.', 'Spectrometer');
```

- *Mission_Phase Table:*

```
-- Sample data for Mission_Phase
INSERT INTO Mission_Phase (Phase_ID, Phase_Name, Start_Date, End_Date, Description, Spacecraft_ID, Launch_Site_ID)
VALUES
  (1, 'Orbital Insertion', '2023-01-15', '2023-01-30', 'Spacecraft enters lunar orbit.', 1, 1),
  (2, 'Lunar Surface Exploration', '2023-02-05', '2023-02-20', 'Rover explores lunar surface.', 2, 1);
```

- *Personnel_Assignment Table (Junction Table for Many-to-Many Relationship):*

```
-- Sample data for Personnel_Assignment
INSERT INTO Personnel_Assignment (Assignment_ID, Personnel_ID, Phase_ID)
VALUES
  (1, 1, 1),
  (2, 2, 2);
```

- *Spacecraft_Instrument Table (Junction Table for Many-to-Many Relationship):*

```
-- Sample data for Spacecraft_Instrument
```

```
INSERT INTO Spacecraft_Instrument (Relationship_ID, Spacecraft_ID, Instrument_ID)
VALUES
  (1, 1, 1),
  (2, 2, 2);
```

Step 4: Design 10 Queries

Certainly! Here are 10 SQL queries that cover different concepts such as SELECT, JOIN, GROUP BY, HAVING, and subqueries for your Chandrayaan 3 mission database:

1. Retrieve a list of all spacecraft used in Chandrayaan 3 :

```
SELECT Name
FROM Spacecraft;
```

2. Find the names of mission personnel involved in the mission:

```
SELECT First_Name, Last_Name
FROM Mission_Personnel;
```

3. Calculate the total cost of the mission:

- Assuming a hypothetical cost column in the `Spacecraft` table.

```
SELECT SUM(Cost) AS Total_Cost
FROM Spacecraft;
```

4. List the instruments used in a specific mission phase (e.g., Phase 1):

```
SELECT i.Name AS Instrument_Name
FROM Instruments i
INNER JOIN Spacecraft_Instrument si ON i.Instrument_ID = si.Instrument_ID
INNER JOIN Mission_Phase mp ON si.Spacecraft_ID = mp.Spacecraft_ID
WHERE mp.Phase_Name = 'Phase 1';
```

5. Find the launch date of Chandrayaan 3:

- Assuming there's only one launch for Chandrayaan 3.

```
SELECT Launch_Date
FROM Launch_Site
WHERE Site_Name = 'Satish Dhawan Space Center';
```

6. List the mission personnel along with the number of phases they are involved in (use GROUP BY and COUNT):

```
SELECT mp.First_Name, mp.Last_Name, COUNT(pa.Phase_ID) AS Phase_Count
FROM Mission_Personnel mp
LEFT JOIN Personnel_Assignment pa ON mp.Personnel_ID = pa.Personnel_ID
```

GROUP BY mp.First_Name, mp.Last_Name;

7. Find the spacecraft that are currently active:
SELECT Name
FROM Spacecraft
WHERE Status = 'Active';

8. Retrieve the mission phases along with the number of instruments used in each phase (use GROUP BY and COUNT):
SELECT mp.Phase_Name, COUNT(si.Relationship_ID) AS Instrument_Count
FROM Mission_Phase mp
LEFT JOIN Spacecraft_Instrument si ON mp.Spacecraft_ID = si.Spacecraft_ID
GROUP BY mp.Phase_Name;
```

*9.Find the average age of mission personnel involved (assuming Date_of_Birth in `Mission_Personnel`):*

*SELECT AVG(YEAR(CURRENT_DATE) - YEAR(Date_of_Birth)) AS Average_Age*
*FROM Mission_Personnel;*

*10. List the mission phases where more than two instruments are used (use HAVING clause):*
*SELECT mp.Phase_Name*
*FROM Mission_Phase mp*
*INNER JOIN Spacecraft_Instrument si ON mp.Spacecraft_ID = si.Spacecraft_ID*
*GROUP BY mp.Phase_Name*
*HAVING COUNT(si.Relationship_ID) > 2;*

*Step 5: Construct One Procedure and One Function*
*Certainly! Here's an example of a stored procedure and a user-defined function for your Chandrayaan 3 mission database:*

*1st Example*
*Stored Procedure: CalculateTotalCost*

*This stored procedure calculates the total cost of a specific mission phase by taking the Phase_ID as input.*

*DELIMITER //*

*CREATE PROCEDURE CalculateTotalCost(IN phaseID INT, OUT totalCost DECIMAL(10, 2))*
*BEGIN*
*SELECT SUM(s.Cost)*
*INTO totalCost*
*FROM Spacecraft s*
```

```
    INNER JOIN Mission_Phase mp ON s.Spacecraft_ID = mp.Spacecraft_ID
    WHERE mp.Phase_ID = phaseID;
END //

DELIMITER ;
```

To use this stored procedure, you can call it and provide a Phase_ID as an argument, and it will return the total cost for that phase.

Example Usage:

```
SET @phaseID = 1; -- Replace with the desired phase ID
CALL CalculateTotalCost(@phaseID, @totalCost);
SELECT @totalCost;
```

User-Defined Function: CalculateAge

This user-defined function calculates the age of mission personnel based on their Date_of_Birth. It takes Date_of_Birth as input and returns the calculated age.

```
DELIMITER //

CREATE FUNCTION CalculateAge(dateOfBirth DATE)
RETURNS INT
BEGIN
    DECLARE age INT;
    SET age = YEAR(CURRENT_DATE) - YEAR(dateOfBirth);
    IF DATE_FORMAT(CURRENT_DATE, '%m%d') < DATE_FORMAT(dateOfBirth, '%m%d') THEN
        SET age = age - 1;
    END IF;
    RETURN age;
END //

DELIMITER ;
```

You can use this function in SQL queries to calculate the age of mission personnel based on their Date_of_Birth.

Example Usage:

```
SELECT First_Name, Last_Name, CalculateAge(Date_of_Birth) AS Age
FROM Mission_Personnel;
```

2nd Example

Stored Procedure:
Let's create a stored procedure that calculates the total cost of a mission phase based on the spacecraft used in that phase.

```
DELIMITER //

CREATE PROCEDURE CalculateMissionPhaseCost(IN phaseName VARCHAR(100), OUT totalCost DECIMAL(10, 2))

BEGIN

   DECLARE spacecraftID INT;


   -- Initialize total cost

   SET totalCost = 0.00;


   -- Get the spacecraft ID for the given phase

   SELECT Spacecraft_ID INTO spacecraftID

   FROM Mission_Phase

   WHERE Phase_Name = phaseName;

      -- Calculate the total cost for the phase

   SELECT SUM(Cost) INTO totalCost

   FROM Spacecraft

   WHERE Spacecraft_ID = spacecraftID;

 END //

DELIMITER ;
```

This procedure takes the phase name as input and calculates the total cost of the spacecraft used in that phase, storing the result in the totalCost output parameter.

```
DELIMITER //
CREATE FUNCTION CalculateAge(dateOfBirth DATE) RETURNS INT
BEGIN
   DECLARE age INT;

   -- Calculate the age based on the date of birth
   SET age = YEAR(CURRENT_DATE()) - YEAR(dateOfBirth);

   RETURN age;
```

```
END //
DELIMITER ;
```

This function takes a dateOfBirth parameter and returns the age of the individual based on the current date.

You can use these stored procedures and functions as follows:

```
-- Calculate the total cost of a mission phase (example)
CALL CalculateMissionPhaseCost('Phase 1', @totalCost);
SELECT @totalCost AS Total_Cost;

-- Calculate the age of mission personnel (example)
SELECT First_Name, Last_Name, CalculateAge(Date_of_Birth) AS Age
FROM Mission_Personnel;
```

Step 6: Enable a Trigger

1st Example

Certainly! Let's create a trigger that logs changes to the spacecraft data whenever an update is made to the Spacecraft table.

Here's how you can create the trigger:

```
DELIMITER //

CREATE TRIGGER LogSpacecraftUpdate

AFTER UPDATE ON Spacecraft

FOR EACH ROW

BEGIN

   INSERT INTO Spacecraft_Log (Spacecraft_ID, Updated_At, Old_Status, New_Status)

   VALUES (OLD.Spacecraft_ID, NOW(), OLD.Status, NEW.Status);

END //

DELIMITER ;
```

In this trigger:

- *LogSpacecraftUpdate is the trigger name.*
- *AFTER UPDATE ON Spacecraft specifies that the trigger should fire after an update operation on the Spacecraft table.*
- *FOR EACH ROW indicates that the trigger will execute once for each row affected by the update.*

- ***Inside the trigger, we use the INSERT INTO statement to insert a new record into a hypothetical Spacecraft_Log table. This record includes the Spacecraft_ID, the timestamp when the update occurred (NOW()), the old status (OLD.Status), and the new status (NEW.Status).***

You would need to create the Spacecraft_Log table to store the log data:

```
CREATE TABLE Spacecraft_Log (
   Log_ID INT AUTO_INCREMENT PRIMARY KEY,
   Spacecraft_ID INT,
   Updated_At TIMESTAMP,
   Old_Status VARCHAR(50),
   New_Status VARCHAR(50)
);
```

Now, whenever an update is made to the Spacecraft table, this trigger will automatically log the old and new status values along with a timestamp in the Spacecraft_Log table, allowing you to track changes to spacecraft data

2nd Example

Certainly! Here's an example of a trigger that logs changes to the spacecraft data whenever an update is made in the Spacecraft table:

```
DELIMITER //

CREATE TRIGGER LogSpacecraftUpdate
AFTER UPDATE ON Spacecraft
FOR EACH ROW
BEGIN
   INSERT INTO Spacecraft_Log (Spacecraft_ID, Old_Status, New_Status, Update_Date)
   VALUES (OLD.Spacecraft_ID, OLD.Status, NEW.Status, NOW());
END;

DELIMITER ;
```

In this trigger:

- ***AFTER UPDATE ON Spacecraft specifies that the trigger should activate after an update operation on the Spacecraft table.***
- ***FOR EACH ROW indicates that the trigger should operate for each row affected by the update.***

Inside the trigger:

- ***We insert a record into the Spacecraft_Log table, capturing the Spacecraft_ID, the old Status, the new Status, and the timestamp of the update using NOW().***

Assuming you have a Spacecraft_Log table like the one created in a previous step, this trigger will log changes to the Status of spacecraft whenever an update occurs in the Spacecraft table.

Example usage:

```
-- Update the Status of a spacecraft

UPDATE Spacecraft

SET Status = 'Inactive'

WHERE Spacecraft_ID = 1;
```

After this update, the trigger will automatically add a log entry in the Spacecraft_Log table, recording the change in status along with the timestamp.

The End