SQL Queries

Insert Operation: Provide an interface for the user to specify some input for the insert operation.

Add new launch request with a given launch system, satellite, launch date.

```
INSERT INTO LaunchRequest
VALUES (
        id = <determined by backend>
        is_approved = false
        launch system = < _______ >
        sat_id = < _______ >
        agency_id = <determined based on username>
        scheduled date = < _______ >
        )
```

Delete Operation : Implement a cascade-on-delete situation. Provide an interface for the user to specify some input for the deletion operation. Based on input, deletion should be performed.

Delete given satellite.

```
DELETE FROM Satellite WHERE id = <___>
```

Update Operation: Provide an interface for the user to specify some input for the update operation.

Update the purpose of a constellation

```
UPDATE Constellation
SET purpose = ____
WHERE name = ______
```

Selection - Create one query of this category and provide an interface for the user to specify the selection conditions to be returned. Example:

SELECT Field_01
FROM Table_01
WHERE Field_02 >= 0

Select all Satellites with a certain orbit type.

```
SELECT id
FROM Satellite
```

WHERE orbit_type == < ______ >

Projection - Create one query of this category and provide an interface for the user to specify the projection conditions to be returned.
Example:
SELECT Field_01
FROM Table_01

Select certain details of each orbit, user selects type, longitude, eccentricity, axis.

```
SELECT orbit_id, <_____>
FROM orbit
```

Join Query : Pick one query of this category, which joins at least two tables and performs a meaningful query, and provide an interface for the user to choose this query (e.g. join the Customers and the Transactions table to find the phone numbers of all customers who has purchased a specific item).

Joins the Constellation and Satellite table to find the purpose of the constellation in which the satellite is located.

```
SELECT satellite.id, constellation.purpose
FROM satellite
INNER JOIN constellation ON satellite.constellation = constellation.name
```

Aggregation query: Pick one queries that require the use of distinct aggregation (min, max, average, or count are all fine).

Finds the the orbit with maximum eccentricity.

```
SELECT MAX(eccentricity) AS HighestEccentricity
FROM orbit
```

Nested aggregation with group-by: Pick one query that finds some aggregated value for each group (e.g. the average number of items purchased per customer).

Find the average number of approved launch requests per agency
```
SELECT Avg(count (isApproved)
FROM launch_request
Group by sat_id
```

Division query: Pick one query of this category and provide an interface for the user to choose this query (e.g. find all the customers who bought all the items).

Find users that track all constellations.

```
SELECT DISTINCT utc.username
FROM user_tracks_constellation utc
WHERE NOT EXISTS
(SELECT c.name FROM constellation c
WHERE NOT EXISTS
(SELECT utc2.username FROM user_tracks_constellation utc2 WHERE utc2.constellation_name =
c.name and utc.username = utc2.username));
```