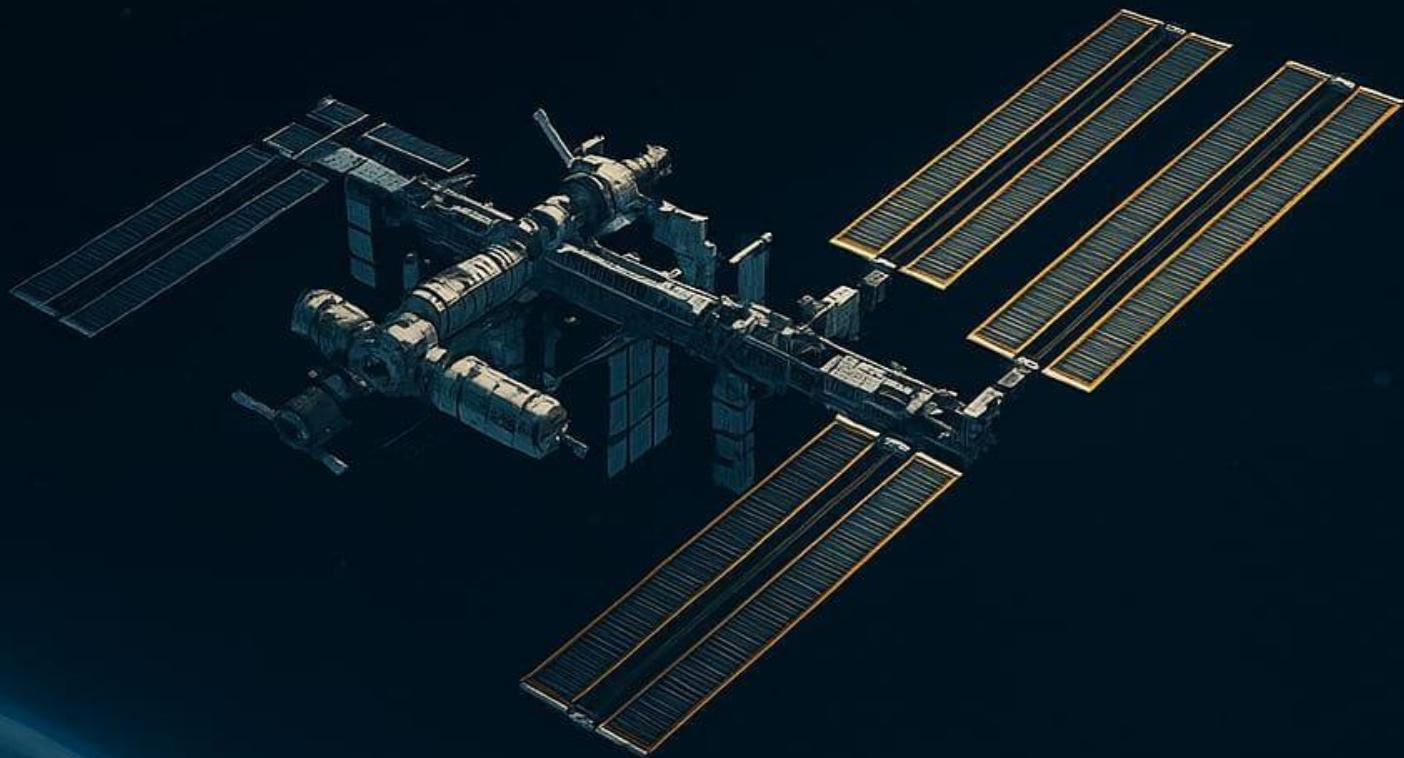


# **INTERNATIONAL SPACE STATION DBMS REPORT**

**CSE 212  
DATABASE SYSTEM LAB**



## **SCHEMA SQUAD**

**ID 23101122 Md Fairuz Anam**

**ID 23101125 Nabiha Afrin**

**ID 23101126 Md Asadullah**

**ID 23101128 Faiaz Masnun Labib**

**Date of Submission:  
6th May 2025**

**Submitted to  
Nadeem Ahmed**

**Assistant Professor, Department of CSE, UAP**

# International Space Station (ISS)

INTERNATIONAL SPACE STATION (ISS)

IT WAS FORMED BY 5 SPACE AGENCIES



US National  
Aeronautics  
and Space  
Administration



The first ISS  
module was  
launched in  
**1998**



Russian  
Federal  
Space Agency



Low-Earth  
orbit **space  
station**



Japan  
Aerospace  
Exploration  
Agency (JAXA)



**Crewed  
habitat**  
artificial  
satellite



European  
Space  
Agency  
(ESA)



Functions  
as a **space  
laboratory**



Canada  
Space  
Agency  
(CSA)



Functions  
as a **space  
laboratory**

Fully orbits the  
Earth in **90  
minutes** (16  
times per day)

Orbital  
altitude at  
**approximately  
400 km**

About as  
**big as a  
soccer  
field**

**Hosts research in  
such fields as**  
physics, biology,  
chemistry,  
physiology,  
meteorology



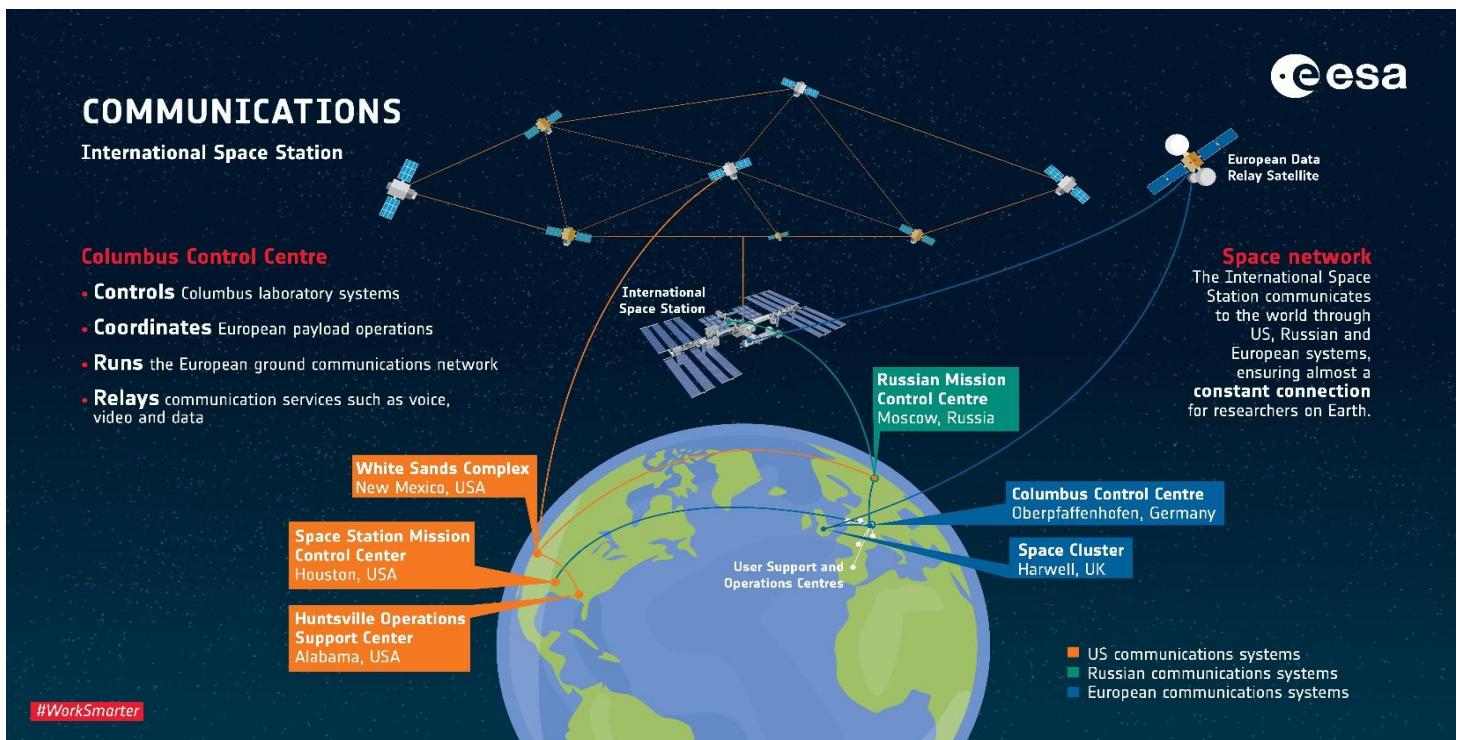
# PROJECT TITLE

## *International Space Station Database Management System (ISS- DBMS)*

### Introduction

The **International Space Station Database Management System (ISS-DBMS)** is a comprehensive digital solution designed to centralize, manage, and optimize the vast and diverse data generated by the operations of the ISS.

This project is built to serve a wide range of users—**administrators, astronauts, mission control staff, and researchers**—by providing a structured platform that ensures **efficient data handling, informed decision-making, and enhanced scientific research**.



### Project Objective

The primary goal of ISS-DBMS is to **simulate a real-world, enterprise-level space operations database** by:

- Organizing mission-critical data into structured tables.
- Enabling powerful and accurate data queries.
- Demonstrating the complex relationships between various operational aspects of the ISS.
- Supporting stakeholder-specific functionalities (e.g., astronauts logging spacewalks, scientists accessing experiment data, doctors reviewing health reports).

## Target Users

- **Administrators:** Manage users, system settings, and permissions.
- **Researchers:** Analyze experimental results and module conditions.
- **Mission Control Personnel:** Oversee operations, crew assignments, supply missions.
- **Astronauts:** Access mission plans, log health data, conduct spacewalk entries.

## Database Entities & Structure

The ISS-DBMS includes **15 interrelated tables (entities)** designed using principles of normalization and real-world database schema standards.

---

### Entity Overview: Purpose & Functional Role

Entity Name	Purpose & Functionality
<b>space_agencies</b>	Stores global space agency details (e.g., NASA, JAXA) to map astronauts, spacecraft, and experiments to their governing agencies.
<b>astronauts</b>	Captures biographical and professional data of crew members participating in space missions.
<b>spacecraft</b>	Contains specs, launch history, and status of vehicles used in ISS missions and supply runs.
<b>experiments</b>	Holds metadata on scientific research activities conducted aboard the ISS, including lead scientists.
<b>modules</b>	Represents the physical sections of the ISS (labs, storage, living units), enabling mapping of equipment, experiments, and environmental data.
<b>mission</b>	Logs operational details of space missions (start/end dates, name, budget), forming a central entity for assignments and activity logs.
<b>crew_assignments</b>	Bridge table connecting astronauts and missions. Tracks who is assigned to what mission.
<b>spacewalks</b>	Stores records of astronaut EVAs (spacewalks), linked to missions and astronauts with durations and purposes.
<b>docking_events</b>	Logs events where spacecraft dock to modules, linking vehicles to ISS components and tracking resupply events.
<b>supply_missions</b>	Contains resupply records, detailing launch dates and cargo, mapped to docking events.
<b>communications</b>	Stores communication setup for each module: frequency and bandwidth used for real-time ops and research.
<b>research_equipment</b>	Tracks scientific tools used during experiments and which module they're stored in or used.
<b>health_records</b>	Logs astronauts' vital statistics and medical observations across missions for health tracking.
<b>doctors</b>	Maps medical personnel to astronauts and specific health entries, linking observations to health logs.
<b>environmental_data</b>	Records environmental readings (temperature, pressure, humidity) within modules to monitor life-support integrity.

## **ER DIAGRAM RELATIONSHIP MAPPING (ORGANIZED)**

### **1. Astronauts ↔ Missions**

**Relationship:** crew\_assignments

**Type:** Many-to-Many

**Purpose:** Tracks which astronaut is assigned to which mission.

**Note:** Implemented via a bridge table.

---

### **2. Astronauts ↔ Spacewalks**

**Relationship:** spacewalks

**Type:** Many-to-Many

**Purpose:** Logs each EVA activity and who performed it under which mission.

### **3. Astronauts ↔ Health Records**

**Relationship:** health\_records

**Type:** One-to-Many

**Purpose:** Each astronaut can have multiple health records during different missions.

---

### **4. Health Records ↔ Doctors**

**Relationship:** doctors

**Type:** One-to-Many

**Purpose:** Doctors are linked to health observations; tracks which doctor reviewed or submitted which record.

---

### **5. Experiments ↔ Modules**

**Relationship:** research\_equipment

**Type:** One-to-One / Many-to-One

**Purpose:** Each experiment takes place in a specific module using specific equipment.

---

### **6. Experiments ↔ Research Equipment**

**Relationship:** research\_equipment

**Type:** One-to-One / One-to-Many

**Purpose:** Associates experiment-specific tools and their storage locations.

## 7. Modules ↔ Environmental Data

**Relationship:** environmental\_data

**Type:** One-to-Many

**Purpose:** Each module has environmental readings recorded periodically.

---

## 8. Modules ↔ Communications

**Relationship:** communications

**Type:** One-to-Many

**Purpose:** Stores communication details for each module.

---

## 9. Modules ↔ Docking Events

**Relationship:** docking\_events

**Type:** One-to-Many

**Purpose:** Tracks which module each spacecraft docks with.

---

## 10. Spacecraft ↔ Docking Events

**Relationship:** docking\_events

**Type:** One-to-Many

**Purpose:** Records each spacecraft's docking history with specific modules.

---

## 11. Docking Events ↔ Supply Missions

**Relationship:** supply\_missions

**Type:** One-to-One

**Purpose:** Each resupply event is linked to a docking event.

---

## 12. Astronauts ↔ Doctors

**Relationship:** doctors

**Type:** One-to-Many (from astronaut to doctor entry)

**Purpose:** Shows which doctor is responsible for which astronaut's care in a given record

---

## ISS-DBMS: ENTITY-ATTRIBUTE CATALOG

### 1. space\_agencies

Stores information about international space agencies.

Attribute	Description	Key Type
agency_id (  )	Unique ID for each agency	Primary Key
name	Name of the space agency	—
country	Country of origin	—
director	Current director of the agency	—

---

### 2. astronauts

Captures astronaut profiles and demographics.

Attribute	Description	Key Type
astronaut_id (  )	Unique identifier for astronauts	Primary Key
name	Full name	—
nationality	Country represented	—
date_of_birth	Date of birth	—
gender	Gender	—

---

### 3. spacecraft

Contains data on spacecraft used in missions.

Attribute	Description	Key Type
spacecraft_id (  )	Unique spacecraft ID	Primary Key
name	Name of the spacecraft	—
manufacturer	Producing company	—
launch_date	Initial launch date	—
status	Active, Inactive, Retired, etc.	—

## 4. experiments

Tracks all scientific experiments onboard ISS.

Attribute	Description	Key Type
experiment_id (  )	Unique experiment ID	Primary Key
name	Title of the experiment	—
description	Overview of the research	—
principal_investigator	Lead researcher	—

---

## 5. modules

Represents ISS modules where activities are conducted.

Attribute	Description	Key Type
module_id (  )	Unique module identifier	Primary Key
name	Module name (e.g., Columbus)	—
description	Function or type of module	—
launch_date	Launch date of module	—

---

## 6. mission

Details for each mission executed aboard or to the ISS.

Attribute	Description	Key Type
mission_id(  )	Unique mission identifier	Primary Key
name	Mission title	—
launch_date	Launch date	—
end_date	Completion date	—
budget	Mission budget (USD)	—

---

## 7. crew\_assignments

Links astronauts to their assigned missions.

Attribute	Description	Key Type
mission_id(  )	Refers to mission	Primary & Foreign Key
astronaut_id (  )	Refers to astronaut	Primary & Foreign Key

## 8. spacewalks

Logs spacewalks performed during missions.

Attribute	Description	Key Type
mission_id (  )	Related mission	Primary & Foreign Key
astronaut_id(  )	Spacewalking astronaut	Primary & Foreign Key
duration	Length of the spacewalk	—
purpose	Objective of the walk	—
spacewalk_date	Date of the activity	—

---

## 9. docking\_events

Tracks each spacecraft docking with ISS modules.

Attribute	Description	Key Type
docking_event_id (  )	Unique docking event ID	Primary Key
spacecraft_id	Docking spacecraft	Foreign Key
module_id	Target ISS module	Foreign Key
docking_date	When the docking occurred	—

---

## 10. supply\_missions

Details resupply missions to the ISS.

Attribute	Description	Key Type
supply_mission_id (  )	Unique supply mission ID	Primary Key
launch_date	Supply launch date	—
cargo_description	Description of delivered cargo	—
docking_event_id	Linked docking event	Foreign Key

---

## 11. communications

Communication specs for each module.

Attribute	Description	Key Type
communication_id (  )	Unique ID	Primary Key
module_id	Target module	Foreign Key
frequency	Comm frequency	—
bandwidth	Comm bandwidth	—

---

## 12. research\_equipment

Equipment associated with experiments.

Attribute	Description	Key Type
equipment_id(  )	Unique equipment ID	Primary Key
experiment_id	Related experiment	Foreign Key
module_id	Module using this equipment	Foreign Key
description	Equipment specifications	—

---

## 13. health\_records

Logs astronauts' health data over time.

Attribute	Description	Key Type
health_record_id (  )	Unique record ID	Primary Key
astronaut_id	Related astronaut	Foreign Key
date	Record date	—
parameter	Metric name (e.g., BP, HR)	—
value	Measured value	—

## 14. doctors

Connects doctor observations with health data.

Attribute	Description	Key Type
doctors_id	Doctor identifier	—
name	Doctor's name	—
observation	Summary or diagnosis	—
health_record_id(  )	Associated health record	Primary & Foreign Key
astronaut_id (  )	Related astronaut	Primary & Foreign Key

---

## 15. environmental\_data

Module environment logs.

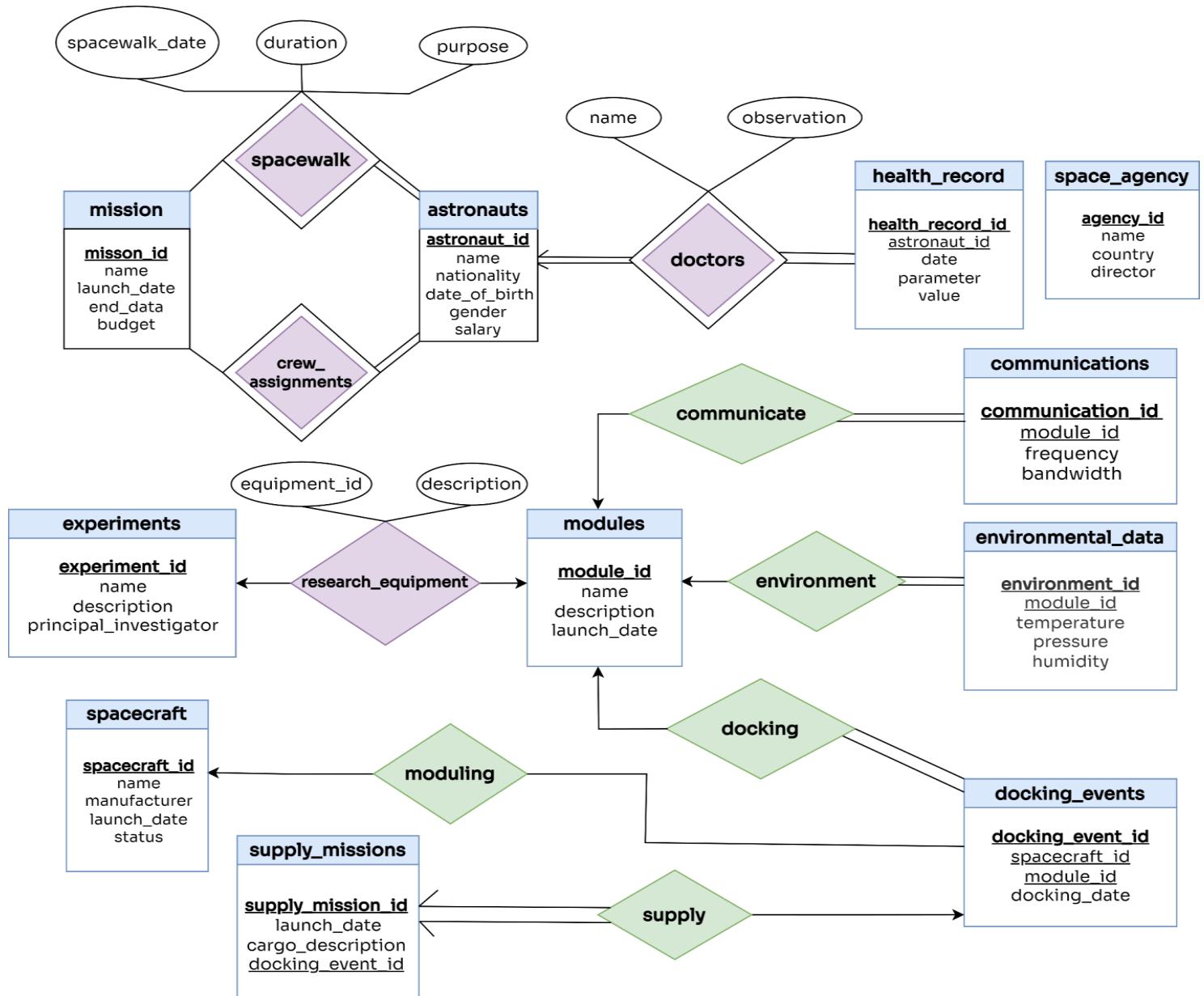
Attribute	Description	Key Type
environment_id (  )	Unique record ID	Primary Key
module_id	Related module	Foreign Key
temperature	Temperature reading	—
pressure	Atmospheric pressure	—
humidity	Humidity level	—

---

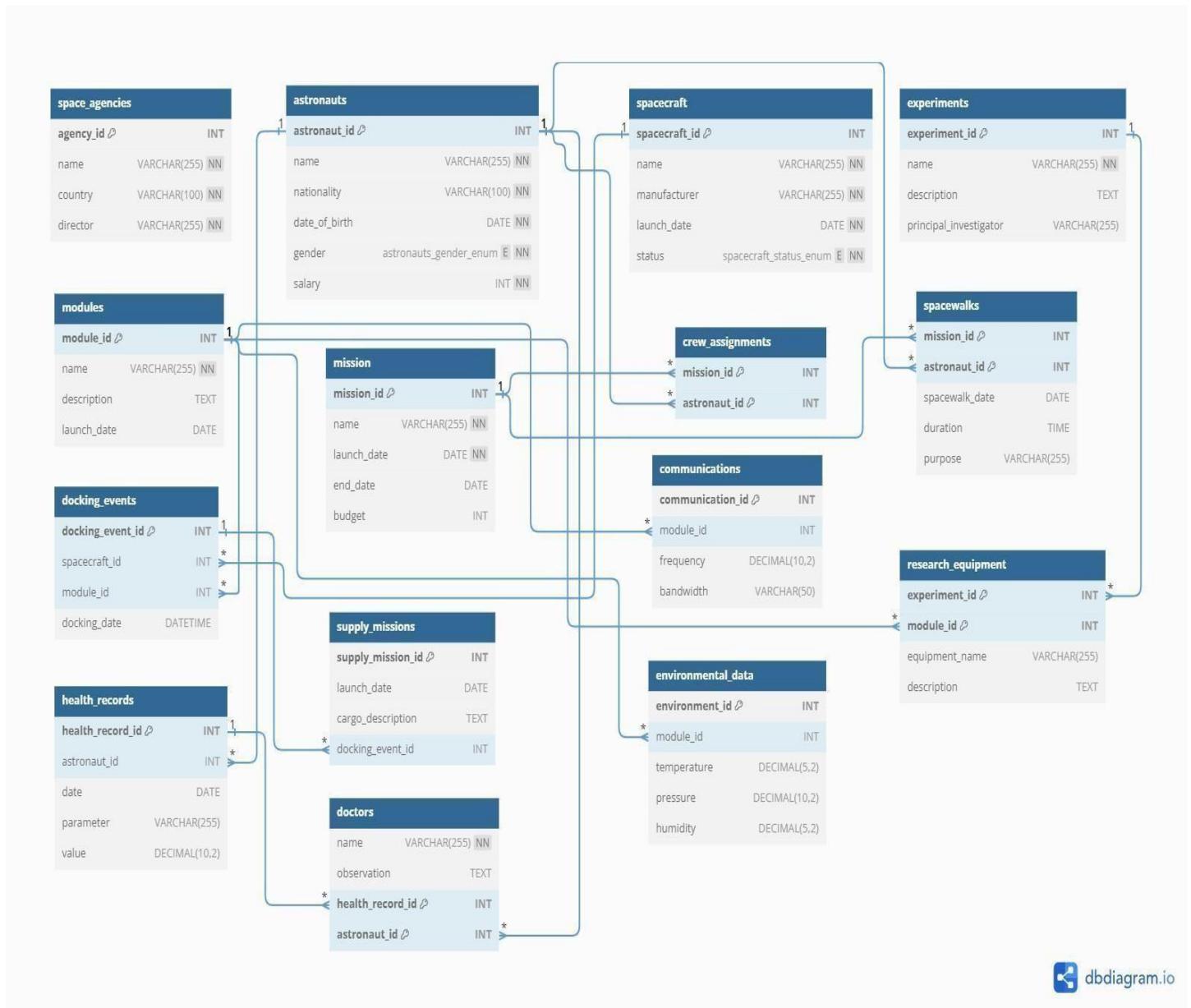
## TECHNOLOGICAL IMPLEMENTATION

- **Database Engine:** Microsoft SQL Server
- **Key Features:**
  - Use of **Primary and Foreign Keys** for integrity.
  - **Normalization** to avoid redundancy.
  - Complex **JOINS, GROUP BY, HAVING**, and **subqueries** implemented.
  - Real-world queries such as tracking salaries over time, evaluating spacewalk durations, monitoring environmental conditions, etc.

# ER DIAGRAM



# SCHEMA DIAGRAM



# QUERY

## 1. Basic Query Structure

**Query:** Finds the names and nationalities of astronauts from the RUSSIA.

```
SELECT name, nationality  
FROM astronauts  
WHERE nationality = 'Russian';
```

### Screenshot

The screenshot shows a SQL query editor interface. On the left, there's a sidebar with a yellow vertical bar and a list titled "Queries:" containing a single item. The main area displays the SQL code:

```
SELECT name, nationality  
FROM astronauts  
WHERE nationality = 'Russian';
```

Below the code, a progress bar indicates ".12 %". At the bottom, there are tabs for "Results" and "Messages", with "Results" being active. The results are displayed in a table:

	name	nationality
1	Alexei Petrov	Russian
2	Anna Sokolova	Russian
3	Sergei Ivanov	Russian
4	Anton Petrov	Russian
5	Vladimir Ivanov	Russian

**Explanation:** Fetches the Names and Nationalities of Astronauts from the RUSSIA.

---

## 2. Additional Basic Operations (Arithmetic)

**Query:** Calculates adjusted annual income of astronauts after monthly tax.

```
SELECT name, (salary - 1000) * 12 AS annual_income  
FROM astronauts;
```

### Screenshot

The screenshot shows a database query execution interface. At the top, a comment is displayed: `-- calculates adjusted annual income of astronauts after monthly tax.`. Below this, the SQL query is shown:

```
SELECT name, (salary - 1000) * 12 AS annual_income  
FROM astronauts;
```

The results are displayed in a table titled "Results". The table has two columns: "name" and "annual\_income". The data consists of 25 rows, each representing an astronaut's name and their calculated annual income after a \$1000 monthly tax deduction. The names are numbered from 1 to 25. The annual incomes range from 108000 to 1308000.

	name	annual_income
1	John Smith	1188000
2	Maria Garcia	1128000
3	Alexei Petrov	1308000
4	Liu Wei	1248000
5	Priya Patel	1068000
6	Yuki Tanaka	1092000
7	Miguel Fernandez	1164000
8	Isabella Rossi	1116000
9	Mohammed Khan	1152000
10	Anna Sokolova	1104000
11	Andrea Müller	1140000
12	Sergei Ivanov	1212000
13	Chen Wei	1080000
14	Rajesh Patel	1176000
15	Yumi Nakamura	1044000
16	Carlos González	1140000
17	Sophia Andrade	1104000
18	Anton Petrov	1128000
19	Mei Ling	1092000
20	Hiroshi Tanaka	1164000
21	Ravi Singh	1188000
22	Yoko Suzuki	1080000
23	Luisa Costa	1116000
24	Vladimir Ivanov	1152000
25	Xiao Chen	1104000

At the bottom of the interface, a green bar indicates: `Query executed successfully.`

**Explanation:** Calculates Adjusted Annual Income of Astronauts After Monthly Tax.

### 3. Set Operations

**Query:** Lists astronauts from either the Indian or Pakistani (no duplicates).

```
(SELECT name FROM astronauts WHERE nationality = 'Indian')
    UNION
```

```
(SELECT name FROM astronauts WHERE nationality = 'Pakistani');
```

#### Screenshot

The screenshot shows a SQL query editor interface. At the top, there is a code editor window containing the following SQL code:

```
-- Lists astronauts from either the Indian or Pakistani (no duplicates).
(SELECT name FROM astronauts WHERE nationality = 'Indian')
    UNION
(SELECT name FROM astronauts WHERE nationality = 'Pakistani');
```

Below the code editor is a results pane. The title bar of the results pane says "112 %". The results tab is selected, showing a table with one column named "name". The data in the table is:

	name
1	Mohammed Khan
2	Priya Patel
3	Rajesh Patel
4	Ravi Singh

**Explanation:** Lists Astronauts from Either the USA Or Russia (No Duplicates).

---

## 4. Join Operation

**Query:** Show Astronaut Names Along with The Name of The Module They Performed Spacewalks Nearby.

```
SELECT a.name AS astronaut_name, m.name AS module_name
FROM astronauts a
JOIN spacewalks s ON a.astronaut_id = s.astronaut_id
JOIN mission ms ON s.mission_id = ms.mission_id
JOIN docking_events d ON ms.mission_id = d.module_id
JOIN modules m ON d.module_id = m.module_id;
```

### Screenshot

The screenshot shows a SQL query being run in SQL Server Management Studio. The query joins five tables to associate astronauts with specific module names. The results are displayed in a table with two columns: 'astronaut\_name' and 'module\_name'. The data includes 25 rows of astronaut names and their corresponding module names, such as John Smith at Unity Node and Yumi Nakamura at Poisk Mini-Research ...

	astronaut_name	module_name
1	John Smith	Unity Node
2	Maria Garcia	Unity Node
3	Alexei Petrov	Unity Node
4	John Smith	Zvezda Service Module
5	Liu Wei	Zvezda Service Module
6	Priya Patel	Zvezda Service Module
7	Yuki Tanaka	Columbus Laboratory
8	Miguel Fernandez	Columbus Laboratory
9	Isabella Rossi	Columbus Laboratory
10	Mohammed Khan	Kibo Laboratory
11	Anna Sokolova	Kibo Laboratory
12	Andrea Müller	Kibo Laboratory
13	Sergei Ivanov	Destiny Laboratory
14	Chen Wei	Destiny Laboratory
15	Rajesh Patel	Destiny Laboratory
16	Yumi Nakamura	Poisk Mini-Research ...
17	Carlos González	Poisk Mini-Research ...
18	Sophia Andrade	Poisk Mini-Research ...
19	Anton Petrov	Cupola
20	Mei Ling	Cupola
21	Hiroshi Tanaka	Cupola
22	Ravi Singh	Tranquility Node
23	Yoko Suzuki	Tranquility Node
24	Luisa Costa	Tranquility Node
25	John Smith	Unknown Node

### **Explanation:**

Joins Astronauts, Spacewalks, Missions, And Modules to Associate Astronauts with Specific Module Names.

## 5. Set Operation (UNION)

**Query:** Get Names of Astronauts Who Either Participated in Spacewalks or Have Health Records.

```
SELECT name  
FROM astronauts  
WHERE astronaut_id IN (SELECT astronaut_id FROM spacewalks)
```

UNION

```
SELECT name  
FROM astronauts  
WHERE astronaut_id IN (SELECT astronaut_id FROM health_records);
```

### Screenshot

The screenshot shows a SQL query editor with the following content:

```
-- Combines Astronauts Involved In Either Spacewalks OR Health Logs, Eliminating Duplicates.  
SELECT name  
FROM astronauts  
WHERE astronaut_id IN (SELECT astronaut_id FROM spacewalks)  
UNION  
SELECT name  
FROM astronauts  
WHERE astronaut_id IN (SELECT astronaut_id FROM health_records);
```

The results pane displays a table with the following data:

	name
1	Alexei Petrov
2	Andrea Müller
3	Anna Sokolova
4	Anton Petrov
5	Carlos González
6	Chen Wei
7	Hiroshi Tanaka
8	Isabella Rossi
9	John Smith
10	Liu Wei
11	Luisa Costa
12	Maria Garcia
13	Mei Ling
14	Miguel Fernández
15	Mohammed Al-
16	Priya Patel
17	Rajesh Patel
18	Ravi Singh
19	Sergei Ivanov
20	Sophia Andrade
21	Vladimir Ivanov

At the bottom, a message bar indicates: "Query executed successfully." and shows a "DE" icon.

### **Explanation:**

Combines Astronauts Involved in Either Spacewalks or Health Logs, Eliminating Duplicates.

## 6. INTERSECT – Finding Common Entries

**Query:** Find astronauts who have **both** participated in spacewalks **and** have health records.

```
SELECT astronaut_id  
FROM spacewalks
```

**INTERSECT**

```
SELECT astronaut_id  
FROM health_records;
```

### Screenshot

The screenshot shows a SQL query editor interface. The query window contains the following code:

```
--- Find astronauts who have both participated in spacewalks and have health records.  
SELECT astronaut_id  
FROM spacewalks  
  
INTERSECT  
  
SELECT astronaut_id  
FROM health_records;
```

The results window displays a table with one column, "astronaut\_id", containing 24 rows of data:

astronaut_id
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24

A green status bar at the bottom of the results window says "Query executed successfully."

### **Explanation:**

This query returns the `astronaut_id`s of astronauts who are **in both** the `spacewalks` and `health_records` tables — meaning they have performed spacewalks **and** have medical records logged.

## 7. EXCEPT – Finding Exclusive Entries

**Query:** Find astronauts who have **health records** but have **never participated in any spacewalk**.

```
SELECT astronaut_id  
FROM health_records
```

EXCEPT

```
SELECT astronaut_id  
FROM spacewalks;
```

### Screenshot

The screenshot shows a SQL query editor window. The query is:

```
--- Finds astronauts who have health records but have never participated in any spacewalk.  
SELECT astronaut_id  
FROM health_records  
EXCEPT  
SELECT astronaut_id  
FROM spacewalks;
```

The results pane is empty, showing only the column header "astronaut\_id".

### **Explanation:**

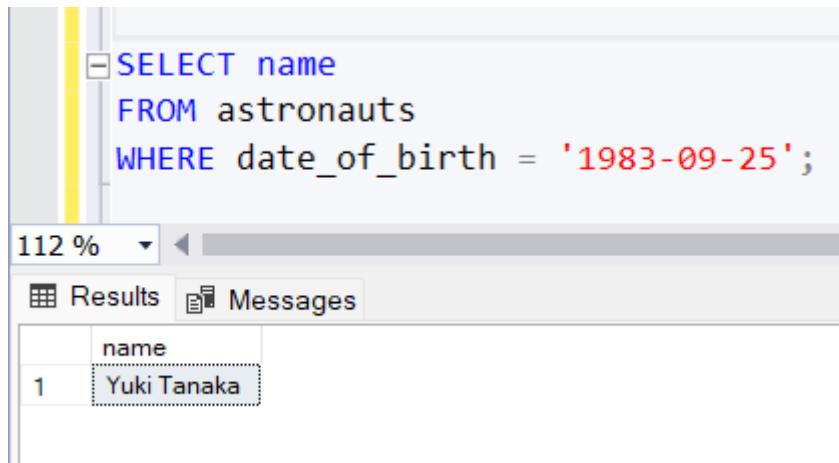
This query returns astronauts who are **only** in the `health_records` table but **not** in the `spacewalks` table — meaning they have never done a spacewalk.

## 8. Birthday Values

**Query:** Find the astronauts with their birthdates.

```
SELECT name  
FROM astronauts  
WHERE date_of_birth = '1983-09-25';
```

### Screenshot



The screenshot shows a database query results window. The query is:

```
SELECT name  
FROM astronauts  
WHERE date_of_birth = '1983-09-25';
```

The results pane shows one row with the name "Yuki Tanaka".

	name
1	Yuki Tanaka

**Explanation:** Finds astronauts with birthdates.

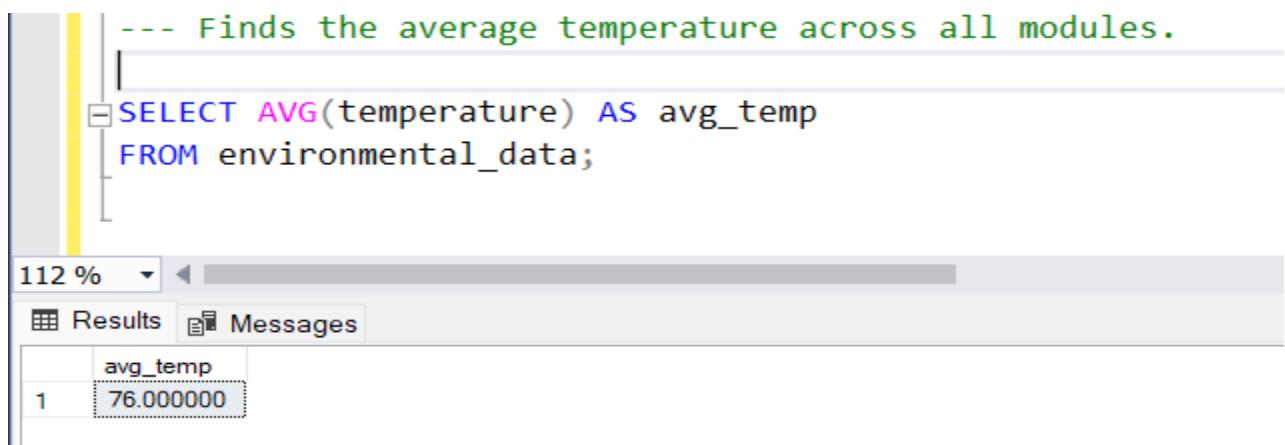
---

## 9. Aggregate Functions

**Query:** Find the average temperature across all modules.

```
SELECT AVG(temperature) AS avg_temp  
FROM environmental_data;
```

### Screenshot



The screenshot shows a database query results window. The query is:

```
--- Finds the average temperature across all modules.  
SELECT AVG(temperature) AS avg_temp  
FROM environmental_data;
```

The results pane shows one row with the value "76.000000".

	avg_temp
1	76.000000

**Explanation:** Returns the average temperature across all modules

## 10. Nested Subqueries

**Query:** Finds Lists astronauts who have done spacewalks longer than 2 hours.

```
SELECT name
FROM astronauts
WHERE astronaut_id IN
(
    SELECT astronaut_id
    FROM spacewalks
    WHERE duration > '02:00:00'
);
```

### Screenshot

The screenshot shows a SQL query editor interface. The query window contains the following code:

```
-- Finds Lists astronauts who have done spacewalks longer than 2 hours.

SELECT name
FROM astronauts
WHERE astronaut_id IN (
    SELECT astronaut_id
    FROM spacewalks
    WHERE duration > '02:00:00'
);
```

The results window displays a table with the following data:

	name
1	John Smith
2	Maria Garcia
3	Alexei Petrov
4	Yumi Nakamura
5	Carlos González
6	Sophia Andrade

**Explanation:** Lists astronauts who have done spacewalks longer than 2 hours.

## 11. Modification – INSERT

**Query:** Helps to insert a new ISS module into the database.

**INSERT INTO** modules (module\_id, name, description, launch\_date)

**VALUES** (101, 'Harmony', 'Utility module', '2007-10-23');

### Screenshot

The screenshot shows a MySQL Workbench interface. In the SQL tab, there is a multi-line comment: --- Inserts a new ISS module into the database. Below it is the SQL statement: INSERT INTO modules (module\_id, name, description, launch\_date) VALUES (101, 'Harmony', 'Utility module', '2007-10-23');. In the Messages tab, the output shows: (1 row affected). Below the messages, the completion time is listed as 2025-05-03T16:27:50.5807582+06:00.

**Explanation:** Inserts a new ISS module into the database.

---

## 12. Modification – DELETE

**Query:** Helps to deletes outdated supply mission records

**DELETE FROM** supply\_missions

**WHERE** launch\_date < '2015-01-01';

### Screenshot

The screenshot shows a MySQL Workbench interface. In the SQL tab, there is a multi-line comment: --- Helps to deletes outdated supply mission records. Below it is the SQL statement: DELETE FROM supply\_missions WHERE launch\_date < '2015-01-01';. In the Messages tab, the output shows: (10 rows affected). Below the messages, the completion time is listed as 2025-05-03T16:30:51.9733257+06:00.

**Explanation:** Deletes outdated supply mission records.

## 13. Modification – UPDATE

**Query:** Helps to Increases temperature by 1 for module with ID 3.

```
UPDATE environmental_data  
SET temperature = temperature + 1  
WHERE module_id = 3;
```

### Screenshot

The screenshot shows a database query execution window. The query is:

```
-- Update & Increases temperature by 1 for module with ID 3 (maybe due to recalibration).  
UPDATE environmental_data  
SET temperature = temperature + 1  
WHERE module_id = 3;
```

The results pane shows:

- 112 % completion
- Messages tab selected
- (1 row affected)
- Completion time: 2025-05-03T16:32:10.2466366+06:00

**Explanation:** Increases temperature by 1 for module with ID 3 (maybe due to recalibration).

---

## 14. String Operations

**Query:** Finds astronauts whose names starts with

```
SELECT name  
FROM astronauts  
WHERE name LIKE 'An%';
```

### Screenshot

The screenshot shows a database query execution window. The query is:

```
-- Finds astronauts whose names starts with "An".  
SELECT name  
FROM astronauts  
WHERE name LIKE 'An%';
```

The results pane shows:

- 112 % completion
- Results tab selected
- Three rows of data:

	name
1	Anna Sokolova
2	Andrea Müller
3	Anton Petrov

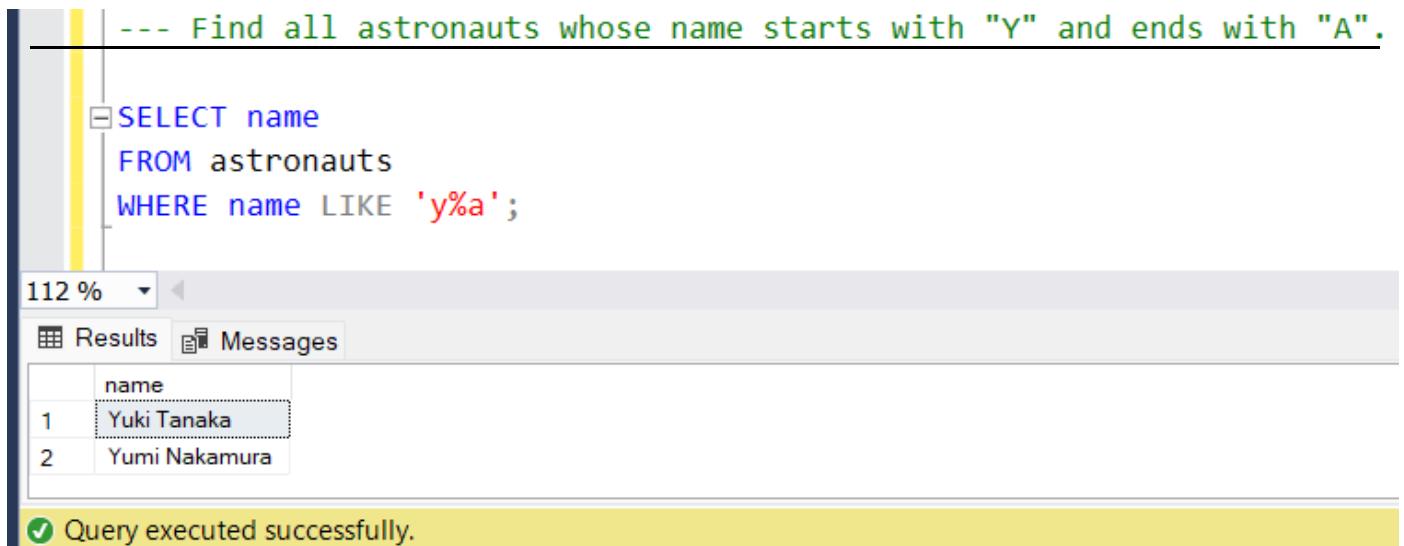
**Explanation:** Finds astronauts whose names contain "An".

## 15. String Matching (LIKE)

**Query:** Find all astronauts whose name starts with "Y" and ends with "A".

```
SELECT name  
FROM astronauts  
  
WHERE name LIKE 'y%a';
```

### Screenshot



The screenshot shows a database query interface. The query window contains the following SQL code:

```
-- Find all astronauts whose name starts with "Y" and ends with "A".  
  
SELECT name  
FROM astronauts  
WHERE name LIKE 'y%a';
```

The results pane displays a table with one column 'name' and two rows:

name
1 Yuki Tanaka
2 Yumi Nakamura

A message bar at the bottom indicates: **Query executed successfully.**

**Explanation:** Uses % as a wildcard to match names starting with "Y" and ending with "A".

## 16. Ordering Results

**Query:** Find astronauts names ascendingly.

```
SELECT name, nationality  
FROM astronauts  
ORDER BY name ASC;
```

### Screenshot

The screenshot shows a database query interface. At the top, there is a text input field containing the query: "SELECT name, nationality FROM astronauts ORDER BY name ASC;". Below the query, a message box displays the instruction: "--- Find astronauts names ascendingly.". The main area shows the results of the query in a table format. The table has two columns: "name" and "nationality". The rows are numbered from 1 to 10. The data is as follows:

	name	nationality
1	Alexei Petrov	Russian
2	Andrea Müller	German
3	Anna Sokolova	Russian
4	Anton Petrov	Russian
5	Carlos González	Spanish
6	Chen Wei	Chinese
7	Hiroshi Tanaka	Japanese
8	Isabella Rossi	Italian
9	John Smith	American
10	Liu Wei	Chinese

At the bottom of the interface, a green success message box states: "Query executed successfully.".

**Explanation:** Lists astronauts alphabetically.

---

## 17. BETWEEN Operator

**Query:** Find astronauts who born between 1980 to 1990

```
SELECT name  
FROM astronauts  
WHERE date_of_birth BETWEEN '1980-01-01' AND '1990-12-31';
```

### Screenshot

The screenshot shows a SQL query execution interface. The query is:

```
--- Find astronauts who born between 1980 to 1990  
SELECT name  
FROM astronauts  
WHERE date_of_birth BETWEEN '1980-01-01' AND '1990-12-31';
```

The results table has a single column 'name' and 18 rows, numbered 1 to 18. The names listed are: John Smith, Maria Garcia, Liu Wei, Priya Patel, Yuki Tanaka, Isabella Rossi, Mohammed Khan, Anna Sokolova, Andrea Müller, Chen Wei, Rajesh Patel, Yumi Nakamura, Sophia Andrade, Anton Petrov, Mei Ling, Hiroshi Tanaka, Yoko Suzuki, Luisa Costa.

A yellow bar at the bottom indicates "Query executed successfully."

**Explanation:** Fetches astronauts born in the 1980s to 1990s.

## 18. GROUP BY

**Query:** Helps to count the astronauts by nationality.

```
SELECT nationality, COUNT(*) AS total_astronauts  
FROM astronauts  
GROUP BY nationality;
```

### Screenshot

The screenshot shows a SQL query being run in a database environment. The query is:

```
--- Helps to count the astronauts by nationality.  
SELECT nationality, COUNT(*) AS total_astronauts  
FROM astronauts  
GROUP BY nationality;
```

The results are displayed in a table:

	nationality	total_astronauts
1	American	1
2	Brazilian	1
3	Chinese	4
4	German	1
5	Indian	3
6	Italian	1
7	Japanese	4
8	Mexican	1
9	Pakistani	1
10	Portugu...	1
11	Russian	5
12	Spanish	2

A message at the bottom indicates the query was executed successfully.

**Explanation:** Counts astronauts by nationality.

---

## 19. GROUP BY with Aggregation

**Query:** Count how many astronauts are assigned to each mission.

```
SELECT mission_id, COUNT(astronaut_id) AS astronaut_count  
FROM crew_assignments  
GROUP BY mission_id;
```

### Screenshot

The screenshot shows a database query editor interface. At the top, there is a comment: `--- Count how many astronauts are assigned to each mission.`. Below it is the SQL query:

```
SELECT mission_id, COUNT(astronaut_id) AS astronaut_count  
FROM crew_assignments  
GROUP BY mission_id;
```

The results section is titled "Results" and displays the following data:

mission_id	astronaut_count
1	3
2	3
3	3
4	3
5	3
6	3
7	3
8	3
9	3
10	3
11	3
12	3
13	3
14	3
15	3

At the bottom of the results pane, a message says "Query executed successfully." with a green checkmark icon.

**Explanation:** Group records by mission and counts how many astronauts are assigned to each.

## 20. HAVING Clause

**Query:** Find nationalities with more than 3 astronauts.

```
SELECT nationality, COUNT(*) AS total  
FROM astronauts  
GROUP BY nationality
```

```
HAVING COUNT(*) > 3;
```

### Screenshot

The screenshot shows a database query interface. The query is:

```
-- Find nationalities with more than 3 astronauts.  
SELECT nationality, COUNT(*) AS total  
FROM astronauts  
GROUP BY nationality  
HAVING COUNT(*) > 3;
```

The results table has two columns: **nationality** and **total**. The data is:

nationality	total
Chinese	4
Japanese	4
Russian	5

(3 rows affected)

Completion time: 2025-05-03T16:53:47.9847479+06:00

At the bottom, a yellow bar indicates: **Query executed successfully.**

**Explanation:** Lists nationalities with more than 3 astronauts.

## 21. Subqueries with IN

**Query:** Find astronauts names who are in mission No. 5

```
SELECT name
FROM astronauts
WHERE astronaut_id IN
(
    SELECT astronaut_id
    FROM crew_assignments
    WHERE mission_id = 5
);
```

### Screenshot

```
-- Find astronauts names who are in mission No. 5

SELECT name
FROM astronauts
WHERE astronaut_id IN
(
    SELECT astronaut_id
    FROM crew_assignments
    WHERE mission_id = 5
);

112 % ▾ Results
name
-----
Sergei Ivanov
Chen Wei
Rajesh Patel

(3 rows affected)

Completion time: 2025-05-03T16:57:28.1259875+06:00

112 % ▾
✔ Query executed successfully.
```

**Explanation:** Gets astronaut names assigned to mission 5.

## 22. Scalar Subquery

**Query:** Find astronaut names with their respective spacewalks numbers.

```
SELECT name,
       (SELECT COUNT(*)
        FROM spacewalks
       WHERE spacewalks.astronaut_id = astronauts.astronaut_id) AS spacewalk_count
      FROM astronauts;
```

### Screenshot

The screenshot shows a MySQL query editor interface. The query window contains the following code:

```
-- Finds astronaut names with their respective number of spacewalks
SELECT name,
       (SELECT COUNT(*)
        FROM spacewalks
       WHERE spacewalks.astronaut_id = astronauts.astronaut_id) AS spacewalk_count
      FROM astronauts;
```

The results window displays the output of the query:

name	spacewalk_count
John Smith	4
Maria Garcia	3
Alexei Petrov	3
Liu Wei	3
Priya Patel	3
Yuki Tanaka	3
Miguel Fernandez	3
Isabella Rossi	3
Mohammed Khan	3
Anna Sokolova	3
Andrea Müller	2
Sergei Ivanov	2
Chen Wei	2
Rajesh Patel	2
Yumi Nakamura	2
Carlos González	2
Sophia Andrade	2
Anton Petrov	2
Mei Ling	2
Hiroshi Tanaka	2
Ravi Singh	2
Yoko Suzuki	2
Luisa Costa	2
Vladimir Ivanov	2
Xiao Chen	2

(25 rows affected)

Query executed successfully.

**Explanation:** Shows astronaut names with their respective number of spacewalks.

## 23. Subquery

**Query:** List all astronauts who have participated in a mission with a budget higher than the average mission budget.

```
SELECT name
FROM astronauts
WHERE astronaut_id IN
(
    SELECT astronaut_id
    FROM crew_assignments
    WHERE mission_id IN
    (
        SELECT mission_id
        FROM mission
        WHERE budget >
        (
            SELECT AVG(CAST(budget AS BIGINT)) FROM mission
        )
    )
);
);
```

### Screenshot

The screenshot shows the SQL Query window in SQL Server Management Studio. The query is as follows:

```
-- list all astronauts who have participated in a mission with a budget higher than the average mission budget.
SELECT name
FROM astronauts
WHERE astronaut_id IN (
    SELECT astronaut_id
    FROM crew_assignments
    WHERE mission_id IN (
        SELECT mission_id
        FROM mission
        WHERE budget > (
            SELECT AVG(CAST(budget AS BIGINT)) FROM mission
        )
    )
);
);
```

The results pane shows the names of 19 astronauts:

name
John Smith
Maria Garcia
Aleksi Petrov
Liu Wei
Priya Patel
Yuki Tanaka
Miguel Fernandes
Isabella Rossi
Mohammed Khan
Anna Sokolova
Andrea Müller
Anton Petrov
Mei Ling
Hirosaki Tanaka
Ravi Singh
Yoko Suzuki
Luisa Costa
Vladimir Ivanov
Xiao Chen

(19 rows affected)

Completion time: 2025-05-08T17:06:26.9554649+06:00

**Explanation:** Nested subqueries are used to filter astronauts who were part of higher-than-average-budget missions.

# PROJECT MAPPING WITH CEP

*Knowledge Profile (K's) Addressed Through Our Project & Mapping Among K's*

K's	Attributes	How K's Are Addressed Through Our Project
K2	Mathematics	Conceptual Mathematics and Numerical Analysis Are Utilized in Various Aspects, Such as Calculating Trajectories, Determining Mission Parameters, Calculating Budgets on Missions, and Analysing Experimental Data.
K3	Engineering Fundamentals	<b>A Systematic Formulation of Engineering Fundamentals is Evident in The Design and Operation of Spacecraft Modules, Experiments, and Communication Systems Within the Database.</b>
K5	Engineering Design	<b>The Project Involves Knowledge Supporting Engineering Design, Exemplified in The Development of The Entity-Relationship (Er) Diagram and Schema Diagram to Represent the Structure of The Database, Facilitating Effective Database Design and Management.</b>
K6	Engineering Practice	<b>Practical Engineering Knowledge Is Applied in Spacecraft Technology, Demonstrated by The Utilization of MySQL Management, and Query Execution, Showcasing Real-World Engineering Practices in Database Development and Deployment.</b>
K8	Research Literature	Engagement With Selected Knowledge from the Research Literature is Reflected in The Integration of Research-Based Findings in Experiment Descriptions, Health Record Analyses, and Equipment Specifications.

## **Complex Engineering Problem (P's) Addressed Through Our Project & Mapping Among P's**

P's	Attributes	How P's Are Addressed Through Our Project
P1	Depth Of Knowledge Required	<b>Depth Of Knowledge Required Is Demonstrated Through the Necessity for In-Depth Understanding of Engineering Fundamentals (K3), Practical Engineering Knowledge (K6), And Engagement with Selected Knowledge from Research Literature (K8), Enabling A Comprehensive Approach to Resolving Complex Issues Like Crew Assignments, Module Functionalities, And Equipment Specifications.</b>
P3	Depth Of Analysis Required	<b>In The Project, Abstract Thinking and Original Analysis Drive the Optimization of Integrity Constraints and Efficiency in Database Management. These Efforts Ensure Robustness and Accuracy, Crucial for Achieving Project Objectives in Space Mission Operations.</b>
P4	Familiarity Of Issues	The Database Tackles Infrequently Encountered Issues Such as Health Record Management, Spacewalk Scheduling, And Equipment Utilization, Reflecting the Complexity of Real-World Space Missions.
P6	Extent Of Stakeholder Involvement & Conflicting Requirements	Diverse Stakeholder Involvement and Conflicting Requirements Are Managed Through the Allocation of Crew Assignments, Coordination of Supply Missions, And Integration of Research Equipment, Reflecting the Multifaceted Nature of Space Exploration.
P7	Interdependence	<b>High-Level Problems with Many Component Parts Are Addressed Through the Management of Spacecraft Systems, Coordination of Missions, And Execution of Experiments, Illustrating the Interdependence of Various Subsystems in Space Operations.</b>

## ***Complex Engineering Activities (A's) Addressed Through our Project & Mapping Among A's***

A's	Attributes	How P's Are Addressed Through Our Project
A1	<b>Range Of Resources</b>	<b>The Project Involves the Utilization of Diverse Resources Such as Astronaut Skills, Spacecraft Equipment, Experiment Materials, And Mission Budgets, Demonstrating the Range of Resources Required for Space Exploration.</b>
A2	<b>Level Of Interaction</b>	<b>Significant Interaction Is Required to Resolve Problems Arising from Spacecraft Operations, Crew Assignments, And Experimental Setups, Highlighting the Complexity of Managing Space Missions.</b>
A3	Innovation	Innovation Is Evident in The Creative Use of Engineering Principles and Research-Based Knowledge to Develop Novel Solutions for Spacecraft Design, Mission Planning, And Experiment Execution.
A5	Familiarity	<b>The Project Extends Beyond Previous Experiences by Applying Principles-Based Approaches to Address Challenges in Space Exploration, Indicating the Project's Contribution to Advancing the Field of Aerospace Engineering.</b>

---