A

Project Report

On

"Space Station Habitat Resource Management"

Developed by

Harvi Gothi	**24cp029**
Pari Patel	**24cp041**

Guided by

Prof. Hemant D. Vasava Sir

Computer Engineering Department

Birla Vishwakarma Mahavidyalaya

Vallabh Vidhyanagar

November 2025

ABSTRACT

The *Space Station Resource Monitoring System* is designed to efficiently manage and monitor critical resources within a space station environment. The project utilizes a relational database system implemented in MySQL, ensuring reliable storage, retrieval, and maintenance of operational data.

The system tracks essential components such as resource levels, consumption logs, medical supplies, and crew activities. Through the use of triggers, the database automatically updates resource quantities and generates alerts when levels drop below defined thresholds. Cursors and stored procedures are used to perform automated checks, such as identifying low resources and expired medical items, thereby enhancing operational safety and efficiency. Additionally, user-defined functions help in determining the real-time status of resources and verifying the validity of supplies.

This project demonstrates the integration of database automation, data integrity enforcement, and decision-support mechanisms within a critical system. By minimizing manual monitoring and ensuring timely updates, the system contributes to maintaining stability and sustainability in the space station's operations.

INDEX

FUNCTIONAL REQUIREMENTS

- The system must allow secure login and assign roles like admin, crew or maintenance.
- The system must allow adding, updating and deleting different types of resources (Oxygen, water, food).
- Each resource must have details like name, type, unit of measurement and whether it is recyclable or not.
- The system must keep track of all the storage units.
- The system must store the details of each crew member.
- The system must keep track of resource consumption on per crew member.
- The system must log each instance when a resource is consumed.
- The system must update resource levels automatically when consumption is recorded.
- The system must track which resources can be recycled and in which units.
- The system must ensure data consistency in across modules.
- The system shall track which crew member is responsible for which maintenance task.
- The system must track the medical supply including stock, usage and expiry dates.

SYSTEM DESIGN

1] Entity Sets

No.	Entity Set	Attributes
1	Resource	ResourceID, Name, Type, UnitOfMeasurement, Recyclable, StorageUnitID
2	StorageUnit	StorageUnitID, Capacity, SupportedResourceTypes
3	ResourceLevel	LevelID, StorageUnitID, ResourceID, CurrentLevel, MinThreshold, MaxThreshold
4	CrewMember	CrewID, Name, Role, AssignedModule, Status, RoleDescription
5	ConsumptionLog	LogID, CrewID, ResourceID, Quantity, DateTime
6	ResupplySchedule	ResupplyID, OrderDate, ExpectedDeliveryDate, ResourceID, Quantity, Status
7	MaintenanceTask	TaskID, CrewID, Description, ScheduledDate, Deadline, Status
8	MedicalSupply	MedicalID, Name, Stock, Usage, ExpiryDate
9	MedicalLog	LogID, CrewID, MedicalID, UsageDetails, DateTime
10	SystemLog	LogID, Type, StatusDetails, DateTime

2] Relationship Sets

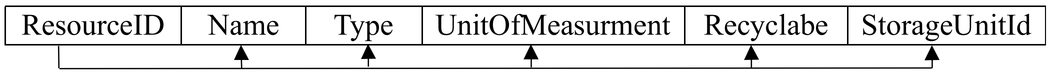
Association	**Relationship Name**	**Mapping Cardinality**
Resource ↔ StorageUnit	Stored_In	One To Many
ResourceLevel ↔ (StorageUnit, Resource)	Monitors	One To Many
CrewMember ↔ ConsumptionLog	Consumes	One To Many
Resource ↔ ResupplySchedule	Recycle	One To Many
ResupplySchedule ↔ Resource	Supplied_By	One To Many
CrewMember ↔ StorageUnit	Assigned_To	Many To One
CrewMember ↔ MaintaenanceTask	Responsible_For	One To Many
CrewMember ↔ Medicallog	Uses_Medical	One To Many

4] Relational Tables

No.	Entity Set	Attributes
1	Resource	ResourceID, Name, Type, UnitOfMeasurement, Recyclable, StorageUnitID
2	StorageUnit	StorageUnitID, Capacity, SupportedResourceTypes
3	ResourceLevel	LevelID, StorageUnitID, ResourceID, CurrentLevel, MinThreshold, MaxThreshold
4	CrewMember	CrewID, Name, Role, AssignedModule, Status, RoleDescription
5	ConsumptionLog	LogID, CrewID, ResourceID, Quantity, DateTime
6	ResupplySchedule	ResupplyID, OrderDate, ExpectedDeliveryDate, ResourceID, Quantity, Status
7	MaintenanceTask	TaskID, CrewID, Description, ScheduledDate, Deadline, Status
8	MedicalSupply	MedicalID, Name, Stock, Usage, ExpiryDate
9	MedicalLog	LogID, CrewID, MedicalID, UsageDetails, DateTime
10	SystemLog	LogID, Type, StatusDetails, DateTime

5] Normalization

1. Resource

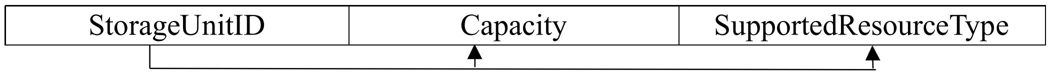


The Candidate key is **ResourceId.**

FD: {ResourceId}$^+ \longrightarrow$ {Name, Type, UnitOfMeasurement, Recyclable, StorageUnitID}

- The table is in 1NF, 2NF (There are no attributes partially dependent on prime attributes), 3NF (There is no transitive dependency), BCNF (Left side contains only super key).

2. StorageUnit

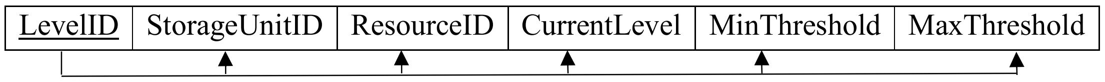


The Candidate key is **StorageUnitID.**

FD: {StorageUnitID}$^+ \longrightarrow$ {Capacity, SupportedResourceTypes}

- SupportedResourceTypes might contain **multiple types** in one field (e.g., “Water, Oxygen”). To make it 1NF-compliant, we separate it into a new table ‘SupportedResource’.
    - SupportedResource(StorageUnitID, ResourceType).
- Now, StorageUnit and SupportedResource are in 1NF, 2NF (There are no attributes partially dependent on prime attributes), 3NF (There is no transitive dependency), BCNF (Left side contains only super key).

3. ResourceLevel

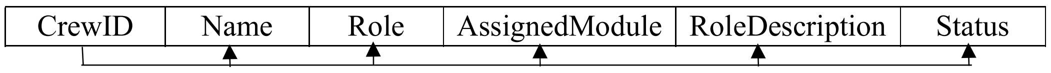


The Candidate key is **LevelID.**

FD: {LevelID}$^+ \longrightarrow$ {StorageUnitID, ResourceID, CurrentLevel, MinThreshold, MaxThreshold}

- The table is in 1NF, 2NF (There are no attributes partially dependent on prime attributes, 3NF (There is no transitive dependency), BCNF (Left side contains only super key).

4. CrewMember



The Candidate key is **CrewID.**

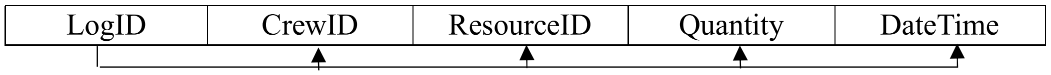
FD: {CrewID}$^+ \longrightarrow$ {Name, Role, AssignedModule, RoleDescription, Status}

- The table is in 1NF, 2NF (There are no attributes partially dependent on prime attributes, but not in 3 NF because RoleDescription depends on Role, not directly on the primary key (CrewID). So, It has transitive dependency. we separate it into a new table ‘RoleInfo’.

  - RoleInfo (Role, RoleDescription)

- Now, CrewMember and RoleInfo are in 1NF, 2NF (There are no attributes partially dependent on prime attributes), 3NF (There is no transitive dependency), BCNF (Left side contains only super key).

5. ConsumptionLog

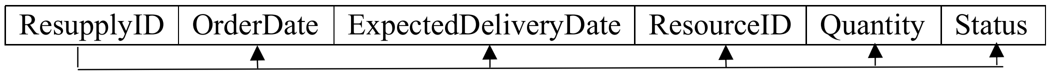


The Candidate key is **LogID.**

FD: $\{LogID\}^{+} \rightarrow$ {LogID, CrewID, ResourceID, Quantity, DateTime}

- The table is in 1NF, 2NF (There are no attributes partially dependent on prime attributes), 3NF (There is no transitive dependency), BCNF (Left side contains only super key).

6. ResupplySchedule

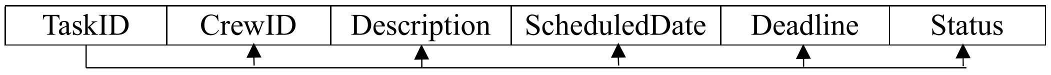


The Candidate key is **ResupplyID.**

FD: $\{ResupplyID\}^{+} \rightarrow$ {ResupplyID, OrderDate, ExpectedDeliveryDate, ResourceID, Quantity, Status}

- The table is in 1NF, 2NF (There are no attributes partially dependent on prime attributes), 3NF (There is no transitive dependency), BCNF (Left side contains only super key).

7. MaintananceTask

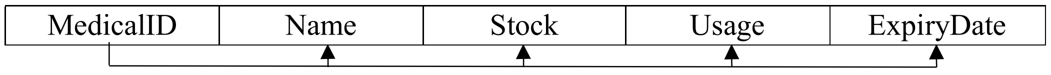


The Candidate key is **TaskID.**

FD: $\{TaskID\}^{+} \rightarrow$ {TaskID, CrewID, Description, ScheduledDate, Deadline, Status}

- The table is in 1NF, 2NF (There are no attributes partially dependent on prime attributes), 3NF (There is no transitive dependency), BCNF (Left side contains only super key).

8. MedicalSupply

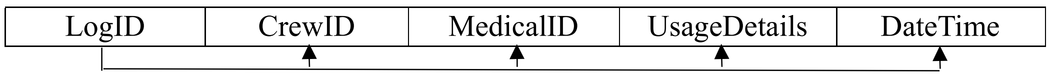


The Candidate key is **MedicalId.**

FD: $\{MedicalID\}^{+} \rightarrow$ {MedicalID, Name, Stock, Usage, ExpiryDate}

- The table is in 1NF, 2NF (There are no attributes partially dependent on prime attributes), 3NF (There is no transitive dependency), BCNF (Left side contains only super key).

9. MedicalLog

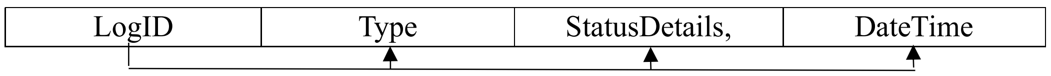


The Candidate key is **LogId.**

FD: $\{LogID\}^{+} \longrightarrow$ {LogID, CrewID, MedicalID, UsageDetails, DateTime}

- The table is in 1NF, 2NF (There are no attributes partially dependent on prime attributes), 3NF (There is no transitive dependency), BCNF (Left side contains only super key).

10. SystemLog



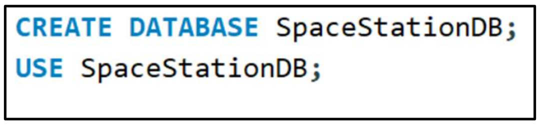
The Candidate key is **LogId.**

FD: $\{LogID\}^{+} \longrightarrow$ {LogID, Type, StatusDetails, DateTime}

- The table is in 1NF, 2NF (There are no attributes partially dependent on prime attributes), 3NF (There is no transitive dependency), BCNF (Left side contains only super key).

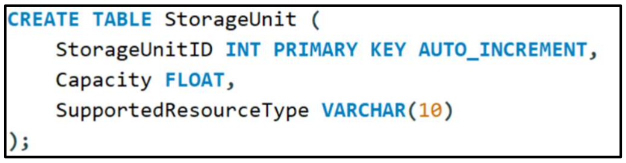
6.Schema

6.1 Create database

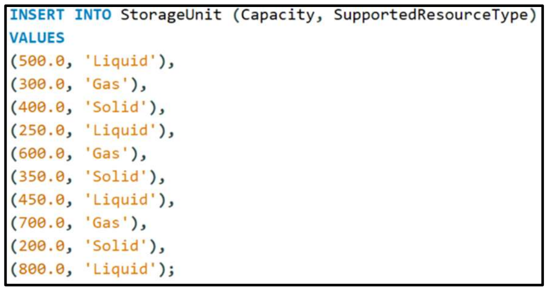


```sql
CREATE DATABASE SpaceStationDB;
USE SpaceStationDB;
```

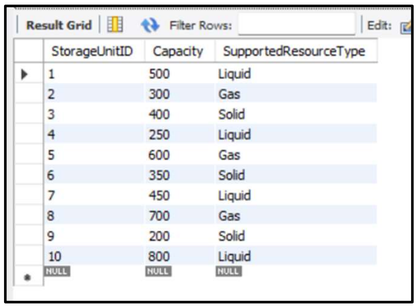
6.2 Storage unit table



```sql
CREATE TABLE StorageUnit (
    StorageUnitID INT PRIMARY KEY AUTO_INCREMENT,
    Capacity FLOAT,
    SupportedResourceType VARCHAR(10)
);
```

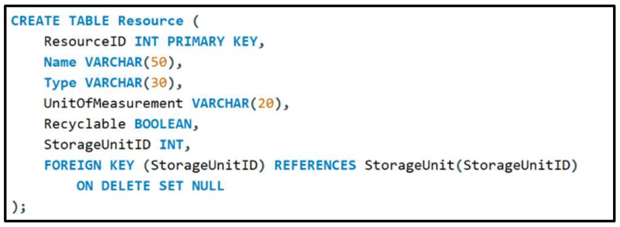


```sql
INSERT INTO StorageUnit (Capacity, SupportedResourceType)
VALUES
(500.0, 'Liquid'),
(300.0, 'Gas'),
(400.0, 'Solid'),
(250.0, 'Liquid'),
(600.0, 'Gas'),
(350.0, 'Solid'),
(450.0, 'Liquid'),
(700.0, 'Gas'),
(200.0, 'Solid'),
(800.0, 'Liquid');
```

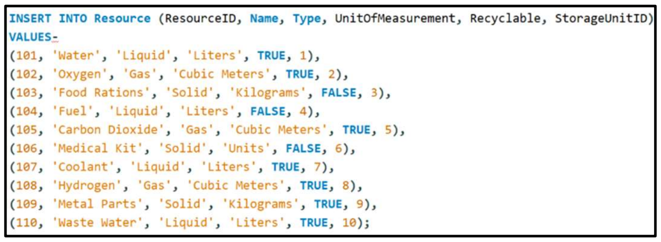


StorageUnitID	Capacity	SupportedResourceType
1	500	Liquid
2	300	Gas
3	400	Solid
4	250	Liquid
5	600	Gas
6	350	Solid
7	450	Liquid
8	700	Gas
9	200	Solid
10	800	Liquid
NULL	NULL	NULL

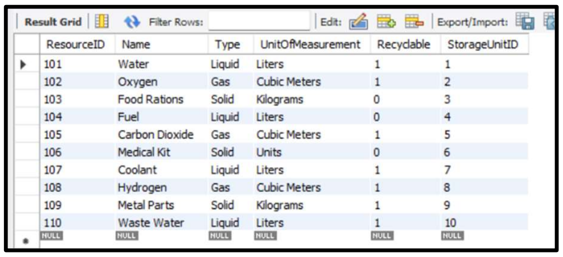
6.3 Resource table



```sql
CREATE TABLE Resource (
    ResourceID INT PRIMARY KEY,
    Name VARCHAR(50),
    Type VARCHAR(30),
    UnitOfMeasurement VARCHAR(20),
    Recyclable BOOLEAN,
    StorageUnitID INT,
    FOREIGN KEY (StorageUnitID) REFERENCES StorageUnit(StorageUnitID)
        ON DELETE SET NULL
);
```

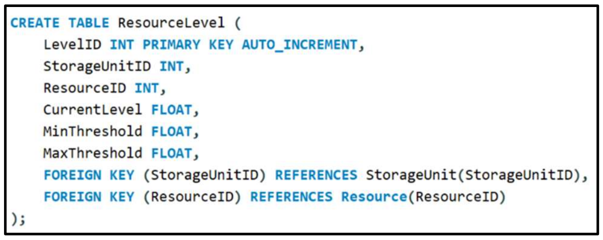


```sql
INSERT INTO Resource (ResourceID, Name, Type, UnitOfMeasurement, Recyclable, StorageUnitID)
VALUES
(101, 'Water', 'Liquid', 'Liters', TRUE, 1),
(102, 'Oxygen', 'Gas', 'Cubic Meters', TRUE, 2),
(103, 'Food Rations', 'Solid', 'Kilograms', FALSE, 3),
(104, 'Fuel', 'Liquid', 'Liters', FALSE, 4),
(105, 'Carbon Dioxide', 'Gas', 'Cubic Meters', TRUE, 5),
(106, 'Medical Kit', 'Solid', 'Units', FALSE, 6),
(107, 'Coolant', 'Liquid', 'Liters', TRUE, 7),
(108, 'Hydrogen', 'Gas', 'Cubic Meters', TRUE, 8),
(109, 'Metal Parts', 'Solid', 'Kilograms', TRUE, 9),
(110, 'Waste Water', 'Liquid', 'Liters', TRUE, 10);
```

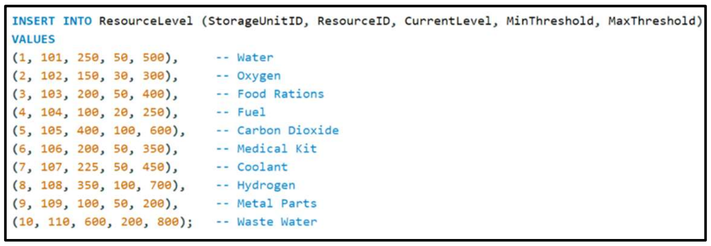


ResourceID	Name	Type	UnitOfMeasurement	Recyclable	StorageUnitID
101	Water	Liquid	Liters	1	1
102	Oxygen	Gas	Cubic Meters	1	2
103	Food Rations	Solid	Kilograms	0	3
104	Fuel	Liquid	Liters	0	4
105	Carbon Dioxide	Gas	Cubic Meters	1	5
106	Medical Kit	Solid	Units	0	6
107	Coolant	Liquid	Liters	1	7
108	Hydrogen	Gas	Cubic Meters	1	8
109	Metal Parts	Solid	Kilograms	1	9
110	Waste Water	Liquid	Liters	1	10
NULL	NULL	NULL	NULL	NULL	NULL

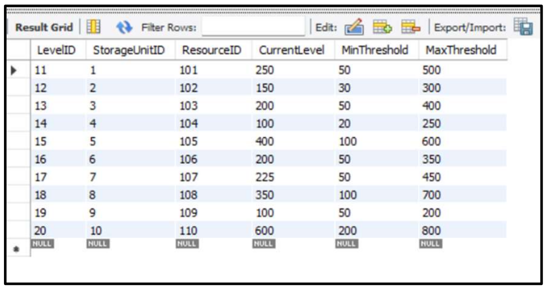
6.4 Resource level



```sql
CREATE TABLE ResourceLevel (
    LevelID INT PRIMARY KEY AUTO_INCREMENT,
    StorageUnitID INT,
    ResourceID INT,
    CurrentLevel FLOAT,
    MinThreshold FLOAT,
    MaxThreshold FLOAT,
    FOREIGN KEY (StorageUnitID) REFERENCES StorageUnit(StorageUnitID),
    FOREIGN KEY (ResourceID) REFERENCES Resource(ResourceID)
);
```

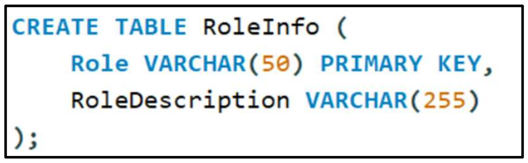


```sql
INSERT INTO ResourceLevel (StorageUnitID, ResourceID, CurrentLevel, MinThreshold, MaxThreshold)
VALUES
(1, 101, 250, 50, 500),     -- Water
(2, 102, 150, 30, 300),     -- Oxygen
(3, 103, 200, 50, 400),     -- Food Rations
(4, 104, 100, 20, 250),     -- Fuel
(5, 105, 400, 100, 600),    -- Carbon Dioxide
(6, 106, 200, 50, 350),     -- Medical Kit
(7, 107, 225, 50, 450),     -- Coolant
(8, 108, 350, 100, 700),    -- Hydrogen
(9, 109, 100, 50, 200),     -- Metal Parts
(10, 110, 600, 200, 800);   -- Waste Water
```

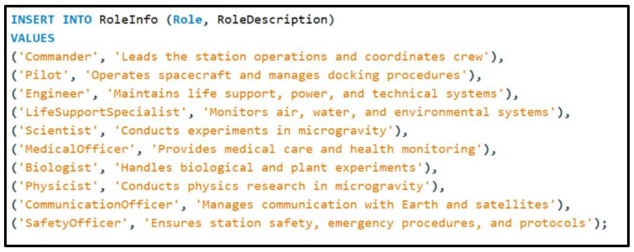


LevelID	StorageUnitID	ResourceID	CurrentLevel	MinThreshold	MaxThreshold
11	1	101	250	50	500
12	2	102	150	30	300
13	3	103	200	50	400
14	4	104	100	20	250
15	5	105	400	100	600
16	6	106	200	50	350
17	7	107	225	50	450
18	8	108	350	100	700
19	9	109	100	50	200
20	10	110	600	200	800
NULL	NULL	NULL	NULL	NULL	NULL

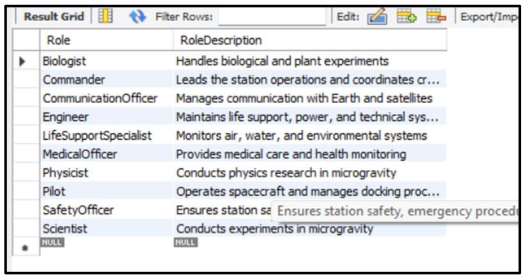
6.5 Role Info



```sql
CREATE TABLE RoleInfo (
    Role VARCHAR(50) PRIMARY KEY,
    RoleDescription VARCHAR(255)
);
```

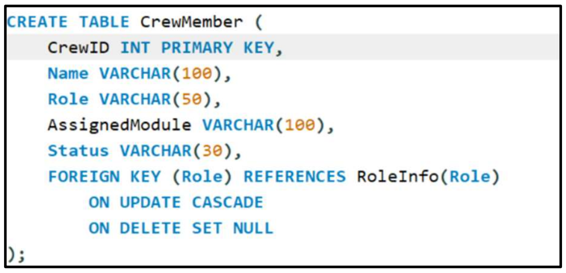


```sql
INSERT INTO RoleInfo (Role, RoleDescription)
VALUES
('Commander', 'Leads the station operations and coordinates crew'),
('Pilot', 'Operates spacecraft and manages docking procedures'),
('Engineer', 'Maintains life support, power, and technical systems'),
('LifeSupportSpecialist', 'Monitors air, water, and environmental systems'),
('Scientist', 'Conducts experiments in microgravity'),
('MedicalOfficer', 'Provides medical care and health monitoring'),
('Biologist', 'Handles biological and plant experiments'),
('Physicist', 'Conducts physics research in microgravity'),
('CommunicationOfficer', 'Manages communication with Earth and satellites'),
('SafetyOfficer', 'Ensures station safety, emergency procedures, and protocols');
```

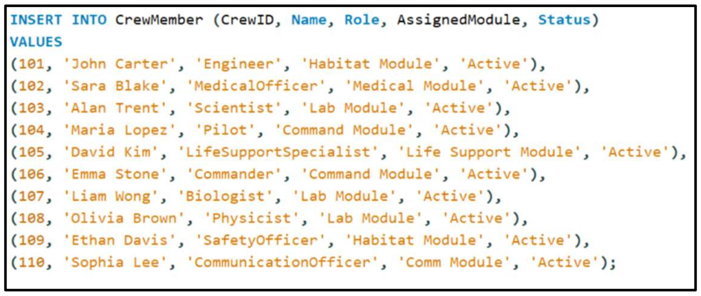


Role	RoleDescription
Biologist	Handles biological and plant experiments
Commander	Leads the station operations and coordinates cr...
CommunicationOfficer	Manages communication with Earth and satellites
Engineer	Maintains life support, power, and technical sys...
LifeSupportSpecialist	Monitors air, water, and environmental systems
MedicalOfficer	Provides medical care and health monitoring
Physicist	Conducts physics research in microgravity
Pilot	Operates spacecraft and manages docking proc...
SafetyOfficer	Ensures station sa Ensures station safety, emergency procedu
Scientist	Conducts experiments in microgravity
NULL	NULL

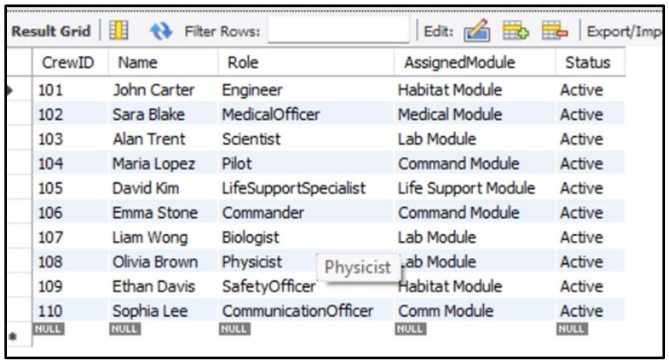
6.6 Crew Member table



```
CREATE TABLE CrewMember (
    CrewID INT PRIMARY KEY,
    Name VARCHAR(100),
    Role VARCHAR(50),
    AssignedModule VARCHAR(100),
    Status VARCHAR(30),
    FOREIGN KEY (Role) REFERENCES RoleInfo(Role)
        ON UPDATE CASCADE
        ON DELETE SET NULL
);
```

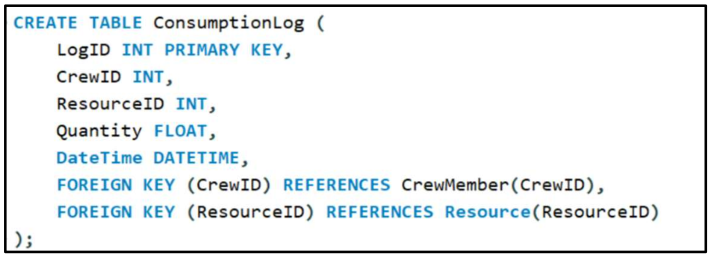


```
INSERT INTO CrewMember (CrewID, Name, Role, AssignedModule, Status)
VALUES
(101, 'John Carter', 'Engineer', 'Habitat Module', 'Active'),
(102, 'Sara Blake', 'MedicalOfficer', 'Medical Module', 'Active'),
(103, 'Alan Trent', 'Scientist', 'Lab Module', 'Active'),
(104, 'Maria Lopez', 'Pilot', 'Command Module', 'Active'),
(105, 'David Kim', 'LifeSupportSpecialist', 'Life Support Module', 'Active'),
(106, 'Emma Stone', 'Commander', 'Command Module', 'Active'),
(107, 'Liam Wong', 'Biologist', 'Lab Module', 'Active'),
(108, 'Olivia Brown', 'Physicist', 'Lab Module', 'Active'),
(109, 'Ethan Davis', 'SafetyOfficer', 'Habitat Module', 'Active'),
(110, 'Sophia Lee', 'CommunicationOfficer', 'Comm Module', 'Active');
```

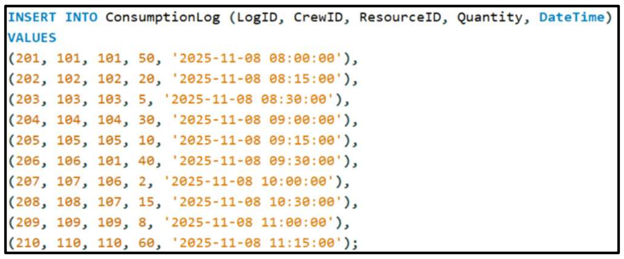


CrewID	Name	Role	AssignedModule	Status
101	John Carter	Engineer	Habitat Module	Active
102	Sara Blake	MedicalOfficer	Medical Module	Active
103	Alan Trent	Scientist	Lab Module	Active
104	Maria Lopez	Pilot	Command Module	Active
105	David Kim	LifeSupportSpecialist	Life Support Module	Active
106	Emma Stone	Commander	Command Module	Active
107	Liam Wong	Biologist	Lab Module	Active
108	Olivia Brown	Physicist	Physicist ab Module	Active
109	Ethan Davis	SafetyOfficer	Habitat Module	Active
110	Sophia Lee	CommunicationOfficer	Comm Module	Active
NULL	NULL	NULL	NULL	NULL

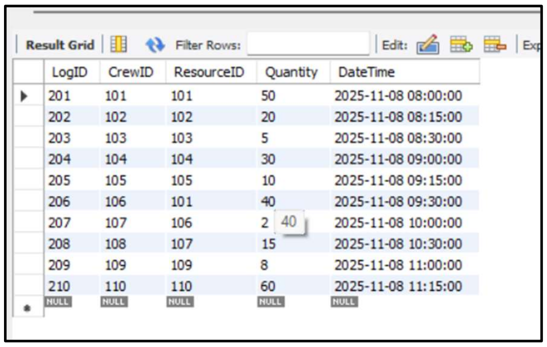
6.7 Consumption log



```
CREATE TABLE ConsumptionLog (
    LogID INT PRIMARY KEY,
    CrewID INT,
    ResourceID INT,
    Quantity FLOAT,
    DateTime DATETIME,
    FOREIGN KEY (CrewID) REFERENCES CrewMember(CrewID),
    FOREIGN KEY (ResourceID) REFERENCES Resource(ResourceID)
);
```

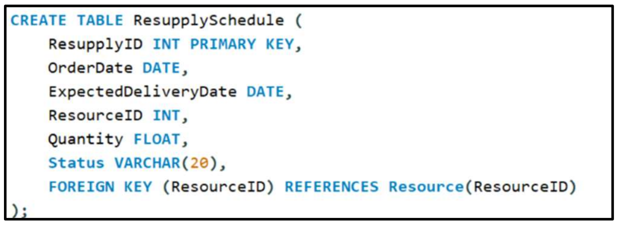


```sql
INSERT INTO ConsumptionLog (LogID, CrewID, ResourceID, Quantity, DateTime)
VALUES
(201, 101, 101, 50, '2025-11-08 08:00:00'),
(202, 102, 102, 20, '2025-11-08 08:15:00'),
(203, 103, 103, 5, '2025-11-08 08:30:00'),
(204, 104, 104, 30, '2025-11-08 09:00:00'),
(205, 105, 105, 10, '2025-11-08 09:15:00'),
(206, 106, 101, 40, '2025-11-08 09:30:00'),
(207, 107, 106, 2, '2025-11-08 10:00:00'),
(208, 108, 107, 15, '2025-11-08 10:30:00'),
(209, 109, 109, 8, '2025-11-08 11:00:00'),
(210, 110, 110, 60, '2025-11-08 11:15:00');
```

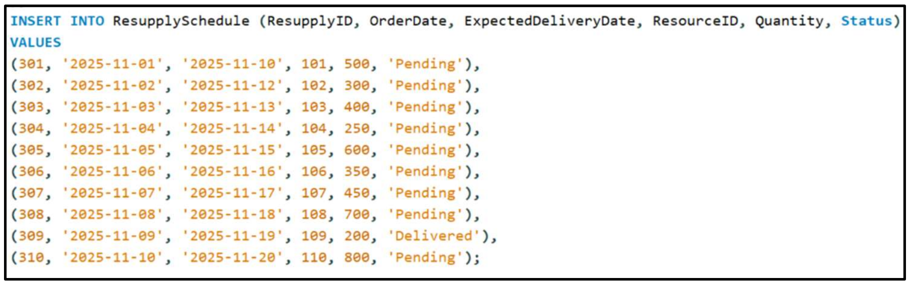


LogID	CrewID	ResourceID	Quantity	DateTime
201	101	101	50	2025-11-08 08:00:00
202	102	102	20	2025-11-08 08:15:00
203	103	103	5	2025-11-08 08:30:00
204	104	104	30	2025-11-08 09:00:00
205	105	105	10	2025-11-08 09:15:00
206	106	101	40	2025-11-08 09:30:00
207	107	106	2	2025-11-08 10:00:00
208	108	107	15	2025-11-08 10:30:00
209	109	109	8	2025-11-08 11:00:00
210	110	110	60	2025-11-08 11:15:00
NULL	NULL	NULL	NULL	NULL

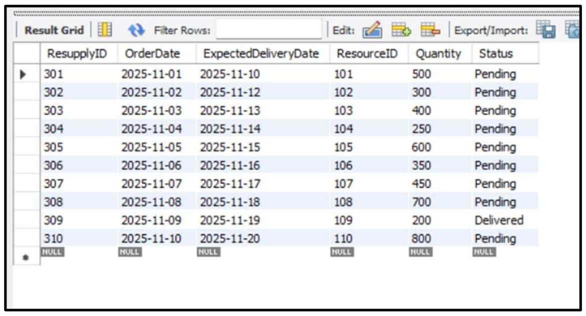
6.8 Resupply schedule table



```sql
CREATE TABLE ResupplySchedule (
    ResupplyID INT PRIMARY KEY,
    OrderDate DATE,
    ExpectedDeliveryDate DATE,
    ResourceID INT,
    Quantity FLOAT,
    Status VARCHAR(20),
    FOREIGN KEY (ResourceID) REFERENCES Resource(ResourceID)
);
```

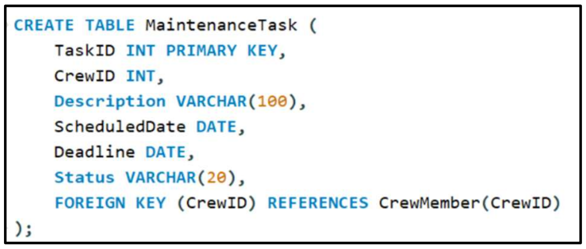


```sql
INSERT INTO ResupplySchedule (ResupplyID, OrderDate, ExpectedDeliveryDate, ResourceID, Quantity, Status)
VALUES
(301, '2025-11-01', '2025-11-10', 101, 500, 'Pending'),
(302, '2025-11-02', '2025-11-12', 102, 300, 'Pending'),
(303, '2025-11-03', '2025-11-13', 103, 400, 'Pending'),
(304, '2025-11-04', '2025-11-14', 104, 250, 'Pending'),
(305, '2025-11-05', '2025-11-15', 105, 600, 'Pending'),
(306, '2025-11-06', '2025-11-16', 106, 350, 'Pending'),
(307, '2025-11-07', '2025-11-17', 107, 450, 'Pending'),
(308, '2025-11-08', '2025-11-18', 108, 700, 'Pending'),
(309, '2025-11-09', '2025-11-19', 109, 200, 'Delivered'),
(310, '2025-11-10', '2025-11-20', 110, 800, 'Pending');
```

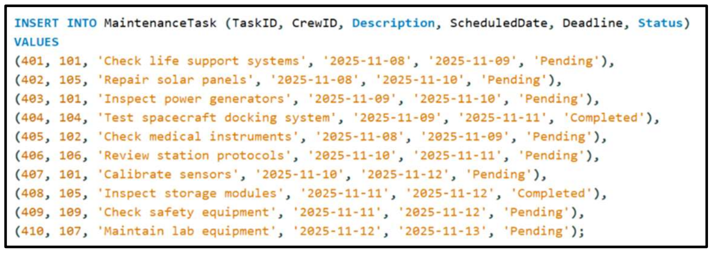


ResupplyID	OrderDate	ExpectedDeliveryDate	ResourceID	Quantity	Status
301	2025-11-01	2025-11-10	101	500	Pending
302	2025-11-02	2025-11-12	102	300	Pending
303	2025-11-03	2025-11-13	103	400	Pending
304	2025-11-04	2025-11-14	104	250	Pending
305	2025-11-05	2025-11-15	105	600	Pending
306	2025-11-06	2025-11-16	106	350	Pending
307	2025-11-07	2025-11-17	107	450	Pending
308	2025-11-08	2025-11-18	108	700	Pending
309	2025-11-09	2025-11-19	109	200	Delivered
310	2025-11-10	2025-11-20	110	800	Pending
NULL	NULL	NULL	NULL	NULL	NULL

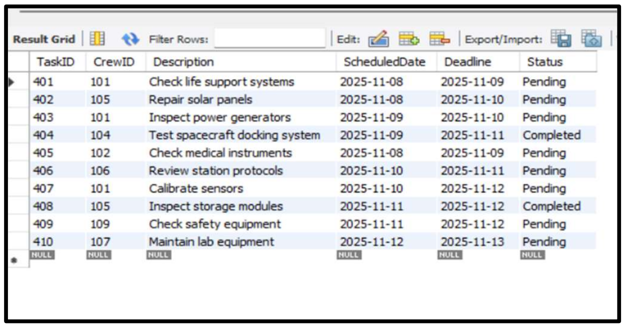
6.9 Maintanenace task



```sql
CREATE TABLE MaintenanceTask (
    TaskID INT PRIMARY KEY,
    CrewID INT,
    Description VARCHAR(100),
    ScheduledDate DATE,
    Deadline DATE,
    Status VARCHAR(20),
    FOREIGN KEY (CrewID) REFERENCES CrewMember(CrewID)
);
```

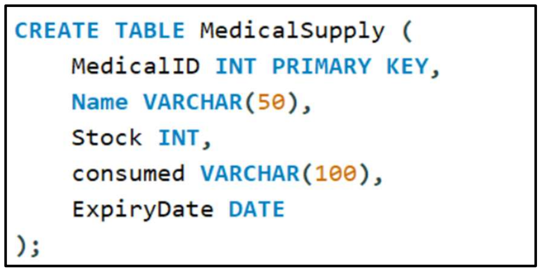


```sql
INSERT INTO MaintenanceTask (TaskID, CrewID, Description, ScheduledDate, Deadline, Status)
VALUES
(401, 101, 'Check life support systems', '2025-11-08', '2025-11-09', 'Pending'),
(402, 105, 'Repair solar panels', '2025-11-08', '2025-11-10', 'Pending'),
(403, 101, 'Inspect power generators', '2025-11-09', '2025-11-10', 'Pending'),
(404, 104, 'Test spacecraft docking system', '2025-11-09', '2025-11-11', 'Completed'),
(405, 102, 'Check medical instruments', '2025-11-08', '2025-11-09', 'Pending'),
(406, 106, 'Review station protocols', '2025-11-10', '2025-11-11', 'Pending'),
(407, 101, 'Calibrate sensors', '2025-11-10', '2025-11-12', 'Pending'),
(408, 105, 'Inspect storage modules', '2025-11-11', '2025-11-12', 'Completed'),
(409, 109, 'Check safety equipment', '2025-11-11', '2025-11-12', 'Pending'),
(410, 107, 'Maintain lab equipment', '2025-11-12', '2025-11-13', 'Pending');
```

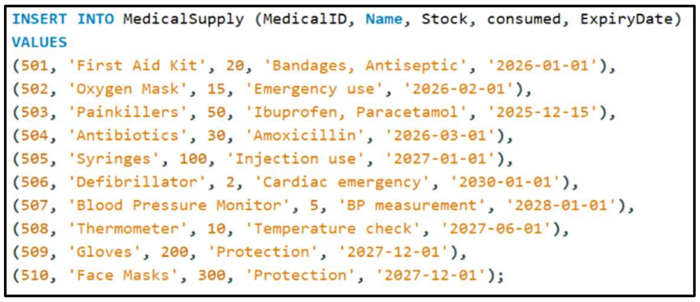


TaskID	CrewID	Description	ScheduledDate	Deadline	Status
401	101	Check life support systems	2025-11-08	2025-11-09	Pending
402	105	Repair solar panels	2025-11-08	2025-11-10	Pending
403	101	Inspect power generators	2025-11-09	2025-11-10	Pending
404	104	Test spacecraft docking system	2025-11-09	2025-11-11	Completed
405	102	Check medical instruments	2025-11-08	2025-11-09	Pending
406	106	Review station protocols	2025-11-10	2025-11-11	Pending
407	101	Calibrate sensors	2025-11-10	2025-11-12	Pending
408	105	Inspect storage modules	2025-11-11	2025-11-12	Completed
409	109	Check safety equipment	2025-11-11	2025-11-12	Pending
410	107	Maintain lab equipment	2025-11-12	2025-11-13	Pending
NULL	NULL	NULL	NULL	NULL	NULL

6.10Medical supply

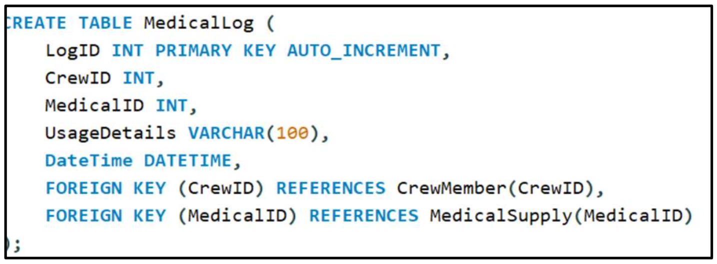


```sql
CREATE TABLE MedicalSupply (
    MedicalID INT PRIMARY KEY,
    Name VARCHAR(50),
    Stock INT,
    consumed VARCHAR(100),
    ExpiryDate DATE
);
```

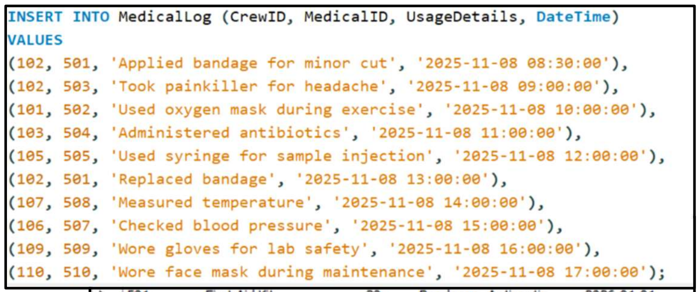


```sql
INSERT INTO MedicalSupply (MedicalID, Name, Stock, consumed, ExpiryDate)
VALUES
(501, 'First Aid Kit', 20, 'Bandages, Antiseptic', '2026-01-01'),
(502, 'Oxygen Mask', 15, 'Emergency use', '2026-02-01'),
(503, 'Painkillers', 50, 'Ibuprofen, Paracetamol', '2025-12-15'),
(504, 'Antibiotics', 30, 'Amoxicillin', '2026-03-01'),
(505, 'Syringes', 100, 'Injection use', '2027-01-01'),
(506, 'Defibrillator', 2, 'Cardiac emergency', '2030-01-01'),
(507, 'Blood Pressure Monitor', 5, 'BP measurement', '2028-01-01'),
(508, 'Thermometer', 10, 'Temperature check', '2027-06-01'),
(509, 'Gloves', 200, 'Protection', '2027-12-01'),
(510, 'Face Masks', 300, 'Protection', '2027-12-01');
```

6.11 Medical Log

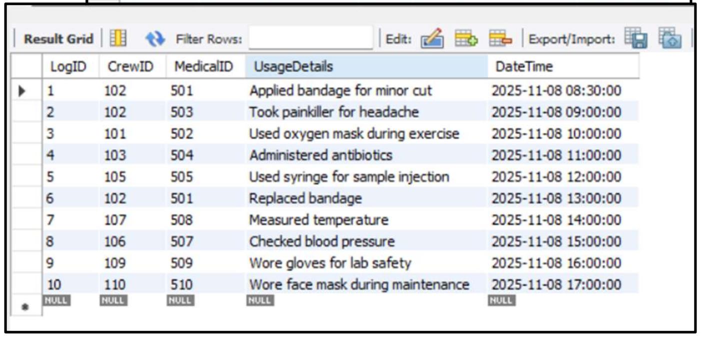


```
REATE TABLE MedicalLog (
    LogID INT PRIMARY KEY AUTO_INCREMENT,
    CrewID INT,
    MedicalID INT,
    UsageDetails VARCHAR(100),
    DateTime DATETIME,
    FOREIGN KEY (CrewID) REFERENCES CrewMember(CrewID),
    FOREIGN KEY (MedicalID) REFERENCES MedicalSupply(MedicalID)
);
```



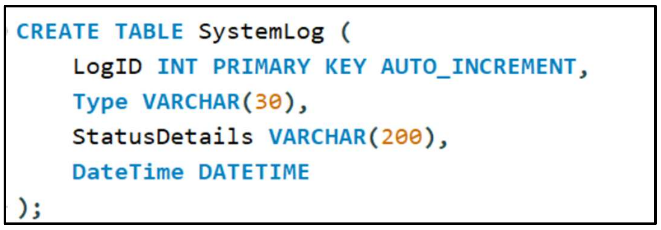
```
INSERT INTO MedicalLog (CrewID, MedicalID, UsageDetails, DateTime)
VALUES
(102, 501, 'Applied bandage for minor cut', '2025-11-08 08:30:00'),
(102, 503, 'Took painkiller for headache', '2025-11-08 09:00:00'),
(101, 502, 'Used oxygen mask during exercise', '2025-11-08 10:00:00'),
(103, 504, 'Administered antibiotics', '2025-11-08 11:00:00'),
(105, 505, 'Used syringe for sample injection', '2025-11-08 12:00:00'),
(102, 501, 'Replaced bandage', '2025-11-08 13:00:00'),
(107, 508, 'Measured temperature', '2025-11-08 14:00:00'),
(106, 507, 'Checked blood pressure', '2025-11-08 15:00:00'),
(109, 509, 'Wore gloves for lab safety', '2025-11-08 16:00:00'),
(110, 510, 'Wore face mask during maintenance', '2025-11-08 17:00:00');
```

```
501    First Aid Kit    20    Bandages, Antiseptic    2026-01-01
```

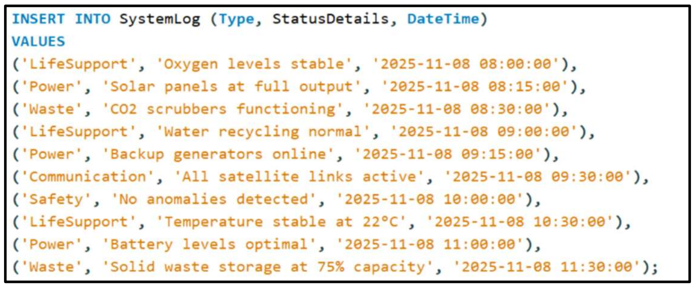


LogID	CrewID	MedicalID	UsageDetails	DateTime
1	102	501	Applied bandage for minor cut	2025-11-08 08:30:00
2	102	503	Took painkiller for headache	2025-11-08 09:00:00
3	101	502	Used oxygen mask during exercise	2025-11-08 10:00:00
4	103	504	Administered antibiotics	2025-11-08 11:00:00
5	105	505	Used syringe for sample injection	2025-11-08 12:00:00
6	102	501	Replaced bandage	2025-11-08 13:00:00
7	107	508	Measured temperature	2025-11-08 14:00:00
8	106	507	Checked blood pressure	2025-11-08 15:00:00
9	109	509	Wore gloves for lab safety	2025-11-08 16:00:00
10	110	510	Wore face mask during maintenance	2025-11-08 17:00:00
NULL	NULL	NULL	NULL	NULL

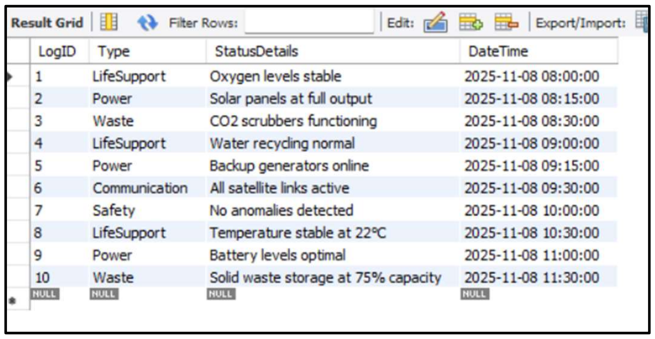
6.12 System Log



```
CREATE TABLE SystemLog (
    LogID INT PRIMARY KEY AUTO_INCREMENT,
    Type VARCHAR(30),
    StatusDetails VARCHAR(200),
    DateTime DATETIME
);
```



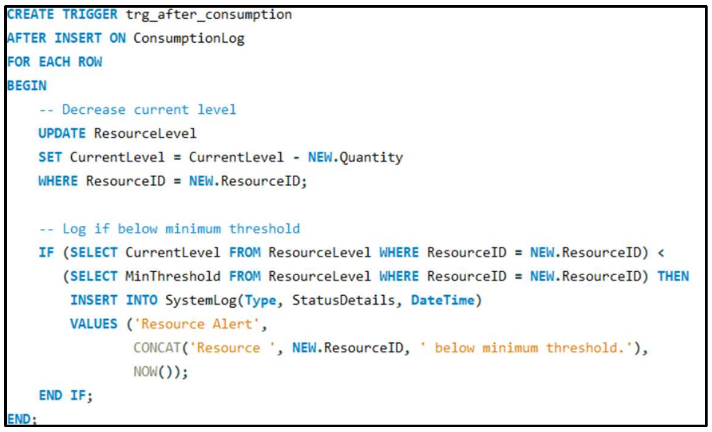
```
INSERT INTO SystemLog (Type, StatusDetails, DateTime)
VALUES
('LifeSupport', 'Oxygen levels stable', '2025-11-08 08:00:00'),
('Power', 'Solar panels at full output', '2025-11-08 08:15:00'),
('Waste', 'CO2 scrubbers functioning', '2025-11-08 08:30:00'),
('LifeSupport', 'Water recycling normal', '2025-11-08 09:00:00'),
('Power', 'Backup generators online', '2025-11-08 09:15:00'),
('Communication', 'All satellite links active', '2025-11-08 09:30:00'),
('Safety', 'No anomalies detected', '2025-11-08 10:00:00'),
('LifeSupport', 'Temperature stable at 22°C', '2025-11-08 10:30:00'),
('Power', 'Battery levels optimal', '2025-11-08 11:00:00'),
('Waste', 'Solid waste storage at 75% capacity', '2025-11-08 11:30:00');
```



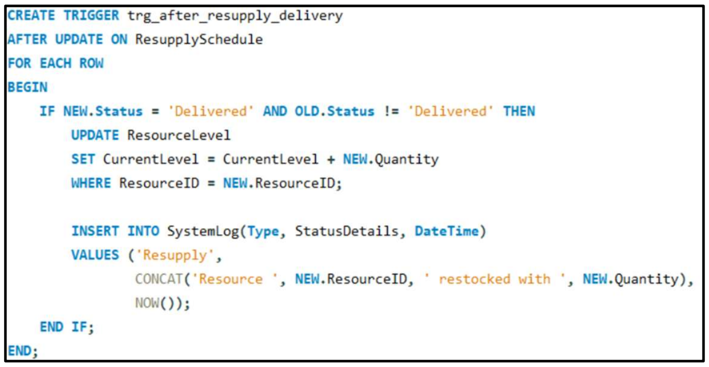
Result Grid | Filter Rows: | Edit: | Export/Import:

LogID	Type	StatusDetails	DateTime
1	LifeSupport	Oxygen levels stable	2025-11-08 08:00:00
2	Power	Solar panels at full output	2025-11-08 08:15:00
3	Waste	CO2 scrubbers functioning	2025-11-08 08:30:00
4	LifeSupport	Water recycling normal	2025-11-08 09:00:00
5	Power	Backup generators online	2025-11-08 09:15:00
6	Communication	All satellite links active	2025-11-08 09:30:00
7	Safety	No anomalies detected	2025-11-08 10:00:00
8	LifeSupport	Temperature stable at 22°C	2025-11-08 10:30:00
9	Power	Battery levels optimal	2025-11-08 11:00:00
10	Waste	Solid waste storage at 75% capacity	2025-11-08 11:30:00
NULL	NULL	NULL	NULL

6.13 Triggers

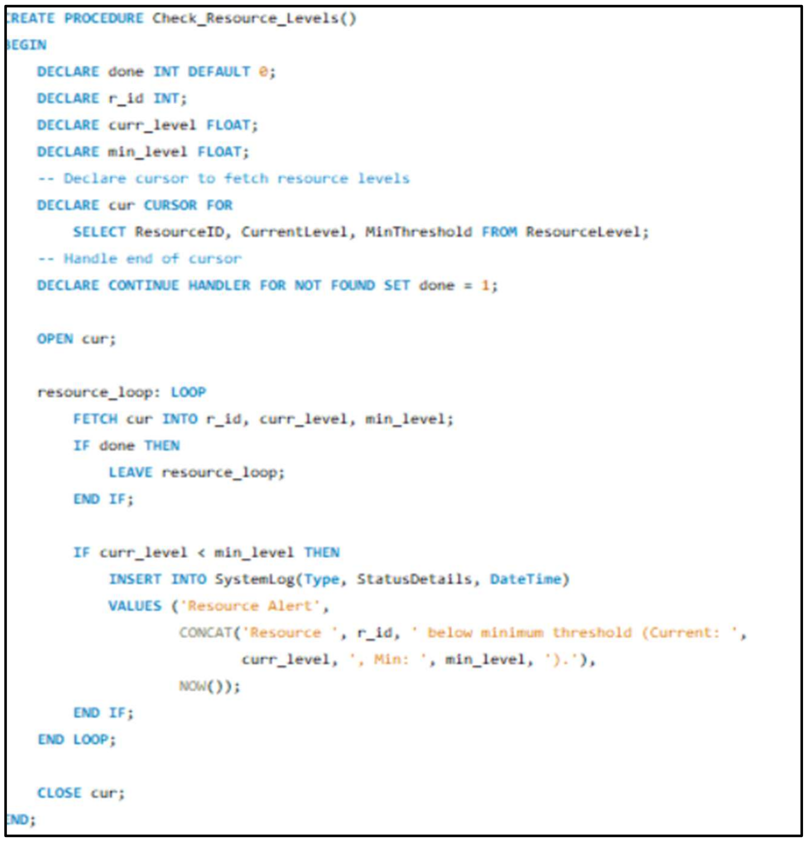


```sql
CREATE TRIGGER trg_after_consumption
AFTER INSERT ON ConsumptionLog
FOR EACH ROW
BEGIN
    -- Decrease current level
    UPDATE ResourceLevel
    SET CurrentLevel = CurrentLevel - NEW.Quantity
    WHERE ResourceID = NEW.ResourceID;

    -- Log if below minimum threshold
    IF (SELECT CurrentLevel FROM ResourceLevel WHERE ResourceID = NEW.ResourceID) <
       (SELECT MinThreshold FROM ResourceLevel WHERE ResourceID = NEW.ResourceID) THEN
        INSERT INTO SystemLog(Type, StatusDetails, DateTime)
        VALUES ('Resource Alert',
                CONCAT('Resource ', NEW.ResourceID, ' below minimum threshold.'),
                NOW());
    END IF;
END;
```

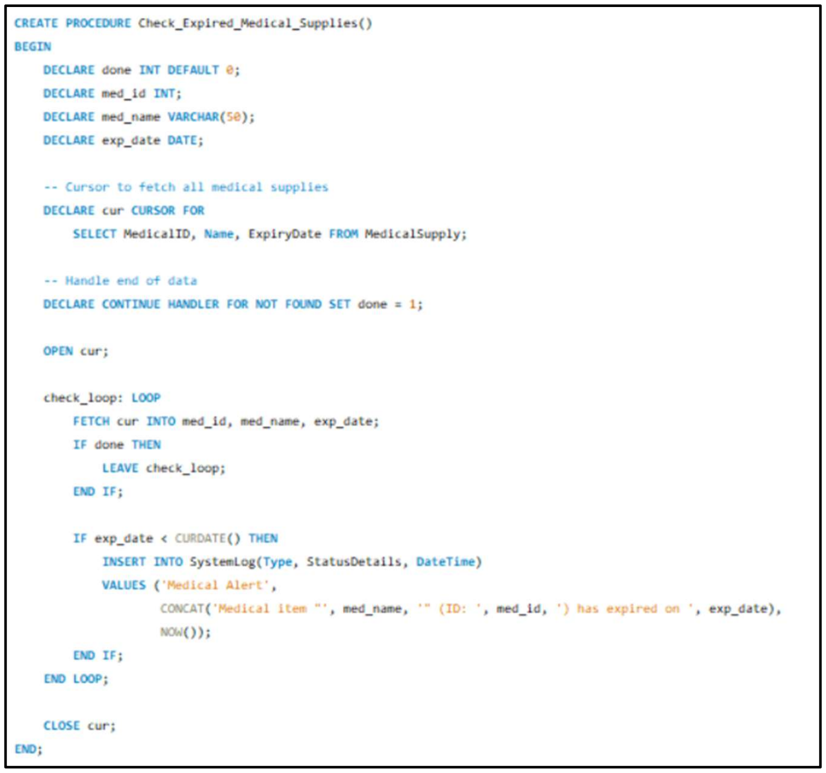


```sql
CREATE TRIGGER trg_after_resupply_delivery
AFTER UPDATE ON ResupplySchedule
FOR EACH ROW
BEGIN
    IF NEW.Status = 'Delivered' AND OLD.Status != 'Delivered' THEN
        UPDATE ResourceLevel
        SET CurrentLevel = CurrentLevel + NEW.Quantity
        WHERE ResourceID = NEW.ResourceID;

        INSERT INTO SystemLog(Type, StatusDetails, DateTime)
        VALUES ('Resupply',
                CONCAT('Resource ', NEW.ResourceID, ' restocked with ', NEW.Quantity),
                NOW());
    END IF;
END;
```

6.14 Procedures and cursors

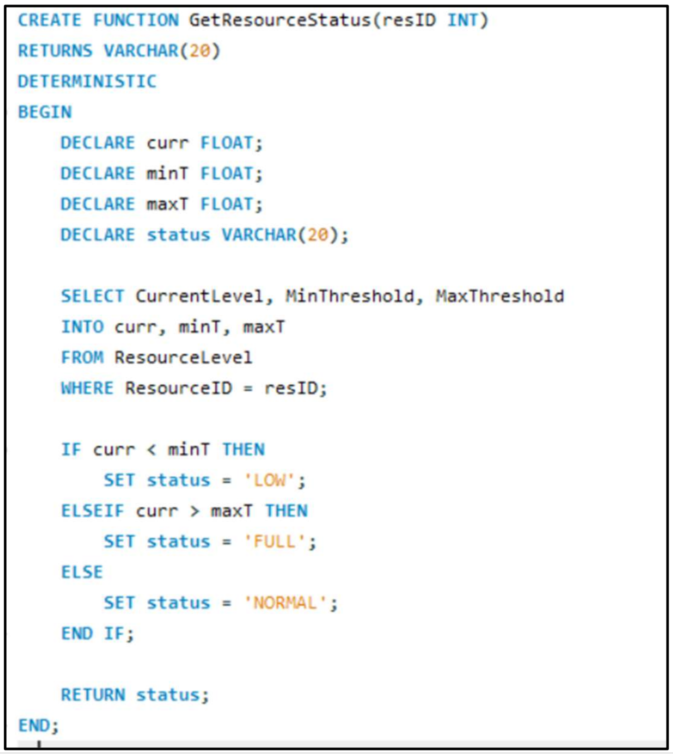


```sql
REATE PROCEDURE Check_Resource_Levels()
EGIN
    DECLARE done INT DEFAULT 0;
    DECLARE r_id INT;
    DECLARE curr_level FLOAT;
    DECLARE min_level FLOAT;
    -- Declare cursor to fetch resource levels
    DECLARE cur CURSOR FOR
        SELECT ResourceID, CurrentLevel, MinThreshold FROM ResourceLevel;
    -- Handle end of cursor
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN cur;

    resource_loop: LOOP
        FETCH cur INTO r_id, curr_level, min_level;
        IF done THEN
            LEAVE resource_loop;
        END IF;

        IF curr_level < min_level THEN
            INSERT INTO SystemLog(Type, StatusDetails, DateTime)
            VALUES ('Resource Alert',
                    CONCAT('Resource ', r_id, ' below minimum threshold (Current: ',
                           curr_level, ', Min: ', min_level, ').'),
                    NOW());
        END IF;
    END LOOP;

    CLOSE cur;
ND;
```

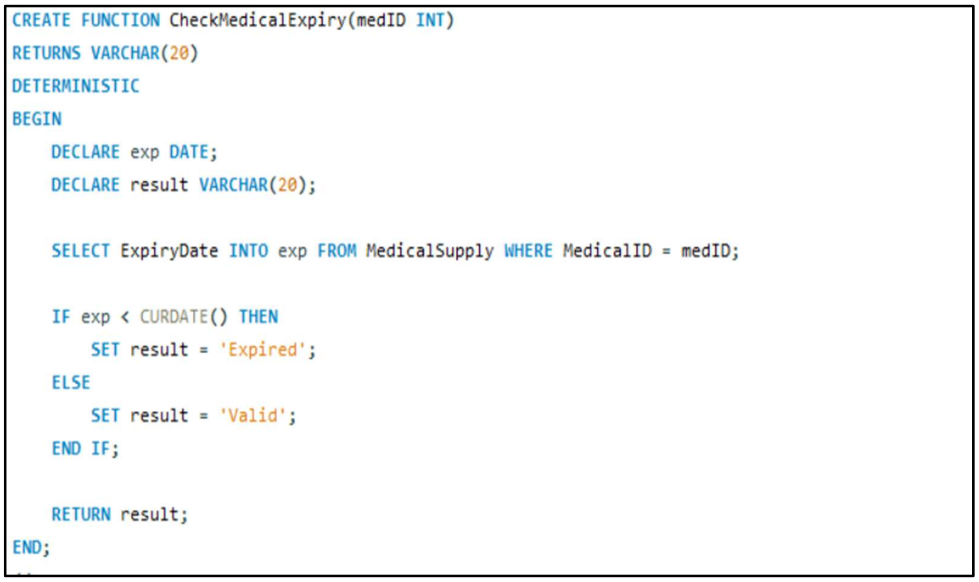


```sql
CREATE PROCEDURE Check_Expired_Medical_Supplies()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE med_id INT;
    DECLARE med_name VARCHAR(50);
    DECLARE exp_date DATE;

    -- Cursor to fetch all medical supplies
    DECLARE cur CURSOR FOR
        SELECT MedicalID, Name, ExpiryDate FROM MedicalSupply;

    -- Handle end of data
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN cur;

    check_loop: LOOP
        FETCH cur INTO med_id, med_name, exp_date;
        IF done THEN
            LEAVE check_loop;
        END IF;

        IF exp_date < CURDATE() THEN
            INSERT INTO SystemLog(Type, StatusDetails, DateTime)
            VALUES ('Medical Alert',
                    CONCAT('Medical item "', med_name, '" (ID: ', med_id, ') has expired on ', exp_date),
                    NOW());
        END IF;
    END LOOP;

    CLOSE cur;
END;
```

6.15 Function

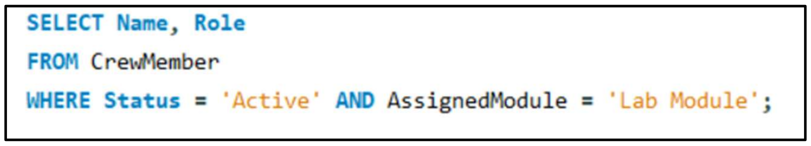


```sql
CREATE FUNCTION GetResourceStatus(resID INT)
RETURNS VARCHAR(20)
DETERMINISTIC
BEGIN
    DECLARE curr FLOAT;
    DECLARE minT FLOAT;
    DECLARE maxT FLOAT;
    DECLARE status VARCHAR(20);

    SELECT CurrentLevel, MinThreshold, MaxThreshold
    INTO curr, minT, maxT
    FROM ResourceLevel
    WHERE ResourceID = resID;

    IF curr < minT THEN
        SET status = 'LOW';
    ELSEIF curr > maxT THEN
        SET status = 'FULL';
    ELSE
        SET status = 'NORMAL';
    END IF;

    RETURN status;
END;
```

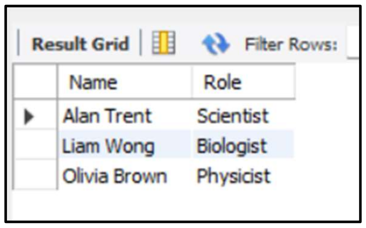


```
CREATE FUNCTION CheckMedicalExpiry(medID INT)
RETURNS VARCHAR(20)
DETERMINISTIC
BEGIN
    DECLARE exp DATE;
    DECLARE result VARCHAR(20);

    SELECT ExpiryDate INTO exp FROM MedicalSupply WHERE MedicalID = medID;

    IF exp < CURDATE() THEN
        SET result = 'Expired';
    ELSE
        SET result = 'Valid';
    END IF;

    RETURN result;
END;
```

Meaningful queries with output

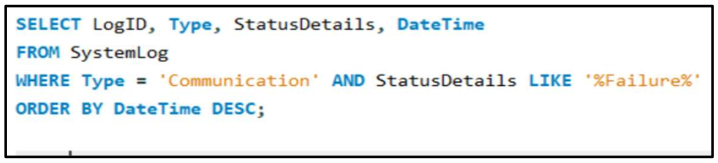


```
SELECT Name, Role
FROM CrewMember
WHERE Status = 'Active' AND AssignedModule = 'Lab Module';
```

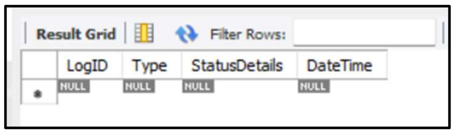


Result Grid
Filter Rows:

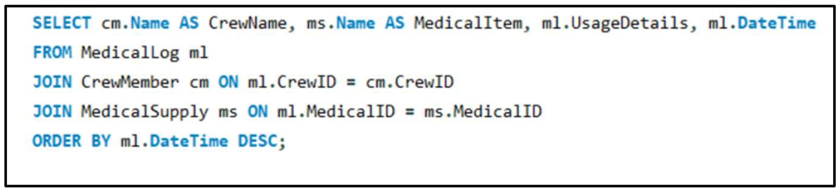
Name	Role
Alan Trent	Scientist
Liam Wong	Biologist
Olivia Brown	Physicist



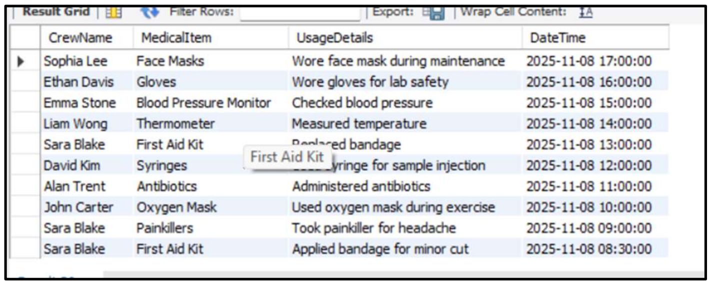
```sql
SELECT LogID, Type, StatusDetails, DateTime
FROM SystemLog
WHERE Type = 'Communication' AND StatusDetails LIKE '%Failure%'
ORDER BY DateTime DESC;
```



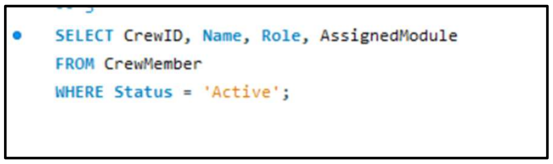
LogID	Type	StatusDetails	DateTime
NULL	NULL	NULL	NULL



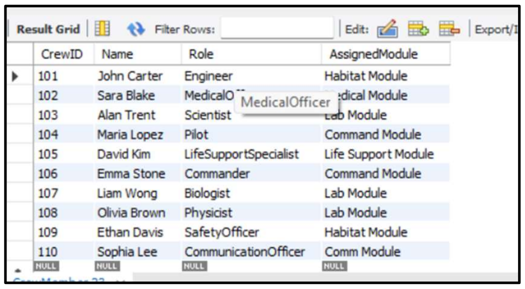
```sql
SELECT cm.Name AS CrewName, ms.Name AS MedicalItem, ml.UsageDetails, ml.DateTime
FROM MedicalLog ml
JOIN CrewMember cm ON ml.CrewID = cm.CrewID
JOIN MedicalSupply ms ON ml.MedicalID = ms.MedicalID
ORDER BY ml.DateTime DESC;
```



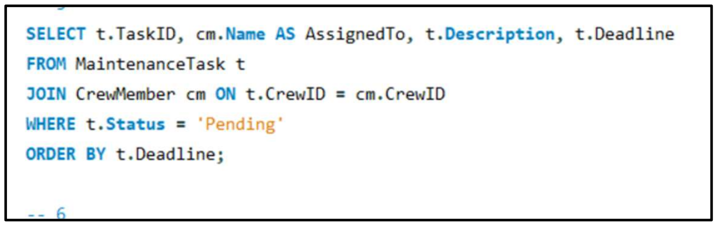
CrewName	MedicalItem	UsageDetails	DateTime
Sophia Lee	Face Masks	Wore face mask during maintenance	2025-11-08 17:00:00
Ethan Davis	Gloves	Wore gloves for lab safety	2025-11-08 16:00:00
Emma Stone	Blood Pressure Monitor	Checked blood pressure	2025-11-08 15:00:00
Liam Wong	Thermometer	Measured temperature	2025-11-08 14:00:00
Sara Blake	First Aid Kit	Replaced bandage	2025-11-08 13:00:00
David Kim	Syringes	[illegible]yringe for sample injection	2025-11-08 12:00:00
Alan Trent	Antibiotics	Administered antibiotics	2025-11-08 11:00:00
John Carter	Oxygen Mask	Used oxygen mask during exercise	2025-11-08 10:00:00
Sara Blake	Painkillers	Took painkiller for headache	2025-11-08 09:00:00
Sara Blake	First Aid Kit	Applied bandage for minor cut	2025-11-08 08:30:00



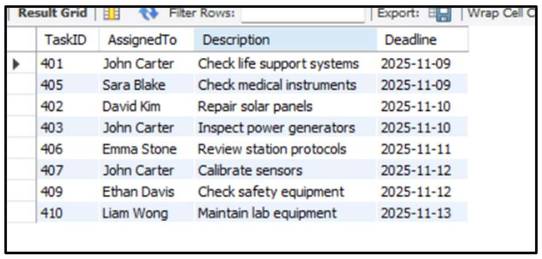
```sql
SELECT CrewID, Name, Role, AssignedModule
FROM CrewMember
WHERE Status = 'Active';
```



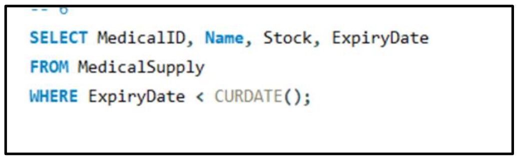
CrewID	Name	Role	AssignedModule
101	John Carter	Engineer	Habitat Module
102	Sara Blake	MedicalOfficer	[illegible]dical Module
103	Alan Trent	Scientist	Lab Module
104	Maria Lopez	Pilot	Command Module
105	David Kim	LifeSupportSpecialist	Life Support Module
106	Emma Stone	Commander	Command Module
107	Liam Wong	Biologist	Lab Module
108	Olivia Brown	Physicist	Lab Module
109	Ethan Davis	SafetyOfficer	Habitat Module
110	Sophia Lee	CommunicationOfficer	Comm Module
NULL	NULL	NULL	NULL



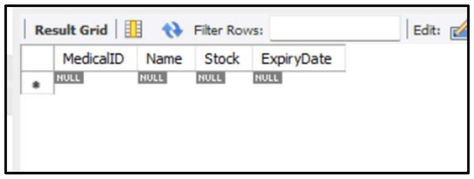
```sql
SELECT t.TaskID, cm.Name AS AssignedTo, t.Description, t.Deadline
FROM MaintenanceTask t
JOIN CrewMember cm ON t.CrewID = cm.CrewID
WHERE t.Status = 'Pending'
ORDER BY t.Deadline;

-- 6
```



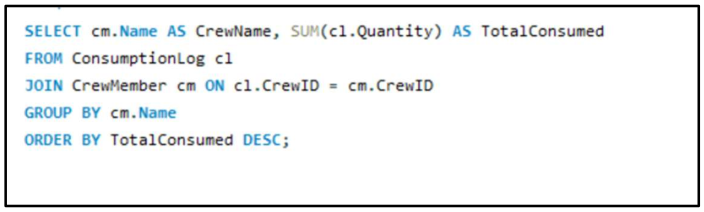
TaskID	AssignedTo	Description	Deadline
401	John Carter	Check life support systems	2025-11-09
405	Sara Blake	Check medical instruments	2025-11-09
402	David Kim	Repair solar panels	2025-11-10
403	John Carter	Inspect power generators	2025-11-10
406	Emma Stone	Review station protocols	2025-11-11
407	John Carter	Calibrate sensors	2025-11-12
409	Ethan Davis	Check safety equipment	2025-11-12
410	Liam Wong	Maintain lab equipment	2025-11-13



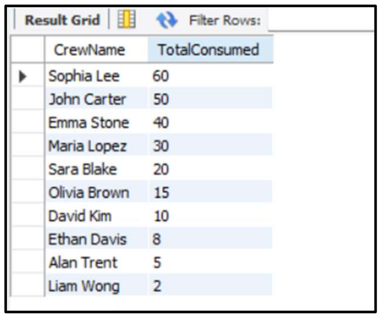
```sql
SELECT MedicalID, Name, Stock, ExpiryDate
FROM MedicalSupply
WHERE ExpiryDate < CURDATE();
```



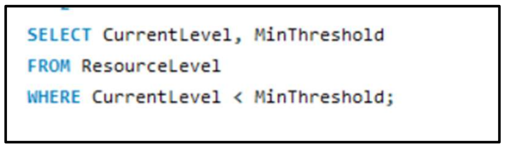
	MedicalID	Name	Stock	ExpiryDate
*	NULL	NULL	NULL	NULL



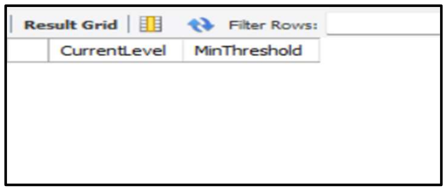
```sql
SELECT cm.Name AS CrewName, SUM(cl.Quantity) AS TotalConsumed
FROM ConsumptionLog cl
JOIN CrewMember cm ON cl.CrewID = cm.CrewID
GROUP BY cm.Name
ORDER BY TotalConsumed DESC;
```



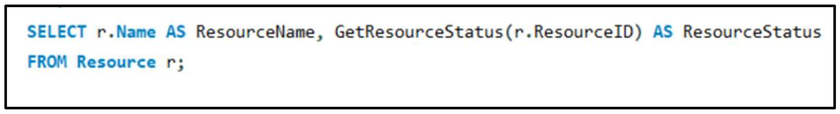
CrewName	TotalConsumed
Sophia Lee	60
John Carter	50
Emma Stone	40
Maria Lopez	30
Sara Blake	20
Olivia Brown	15
David Kim	10
Ethan Davis	8
Alan Trent	5
Liam Wong	2



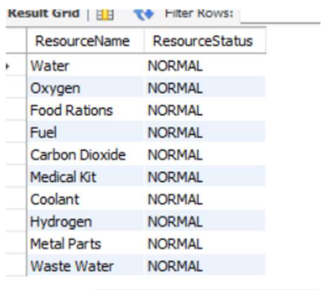
```sql
SELECT CurrentLevel, MinThreshold
FROM ResourceLevel
WHERE CurrentLevel < MinThreshold;
```



CurrentLevel	MinThreshold



```sql
SELECT r.Name AS ResourceName, GetResourceStatus(r.ResourceID) AS ResourceStatus
FROM Resource r;
```



ResourceName	ResourceStatus
Water	NORMAL
Oxygen	NORMAL
Food Rations	NORMAL
Fuel	NORMAL
Carbon Dioxide	NORMAL
Medical Kit	NORMAL
Coolant	NORMAL
Hydrogen	NORMAL
Metal Parts	NORMAL
Waste Water	NORMAL