Application Description

The database that is presented in this report outlines a vaccine delivery system, from a logistical perspective. Within the system, there are the following entities; vaccines, batches, lots, warehouses, employees, delivery vehicles and hospitals. The design ensures all elements of the system are as efficient as possible, whilst maintaining the integrity of the system. To enforce data integrity, several constraints have been introduced, which alleviates batch errors, incorrect categorization of groups of vaccines and the risk of data becoming corrupt.

The core of the system is the vaccine that is to be transported to the designated hospital. Each vaccine shall have a unique identification number, which contains information regarding the dosage of the vaccine, the name of the vaccine, the date the vaccination was manufactured, and the batch number that the vaccine is located in. Each vaccine can only have one batch identification number assigned to it.

The vaccines are stored in groups of batches, by where a unique batch identifier can determine the weight, the warehouse number, the assigned lot number, and the expiration date, and allocation date of the vaccines. Many constraints have been implemented to ensure data integrity within this table.

Lots contain several batches within a warehouse, and have a unique identification number. The lot identifier tells employees where the batches are located within the warehouse. Essentially, the lot table is a locating table. It contains information on which warehouse it is located in, and whether it is available for storage or occupied. Warehouses are divided into lots as mentioned above. The warehouse number identifies the lots available, the total lots, the total number of staff working in the warehouse, and the address of the warehouse.

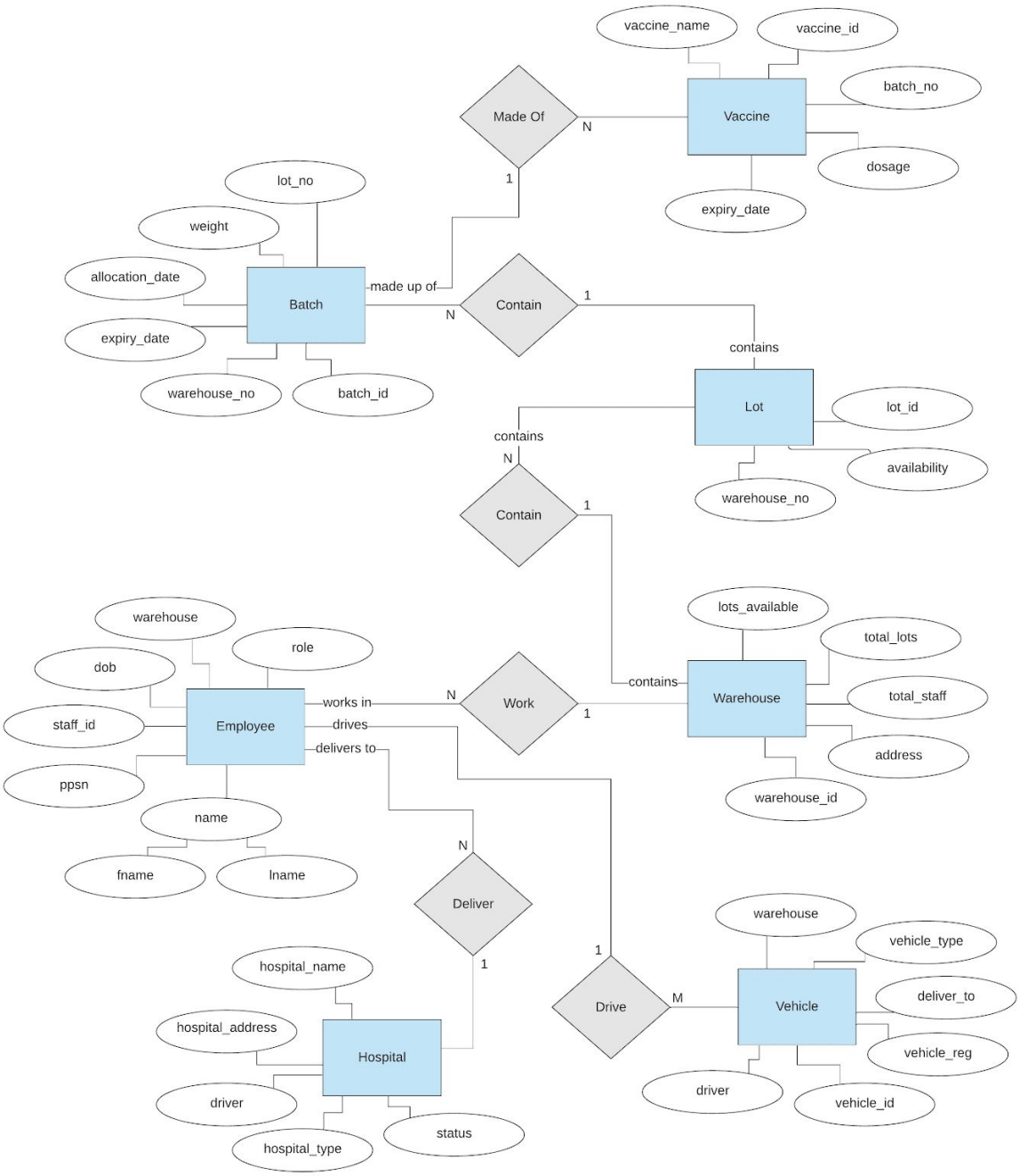
Employees are required to be actively involved in the logistical process, and in this case, the delivery of the batches of vaccines to the hospitals. The staff identification number produces information on the employee such as; their full name, their PPSN, their date of birth, the warehouse in which they are active in, and their role. Additionally, some employees may be drivers, where they will be assigned a delivery vehicle. The vehicle has a unique identification number that is used internally to provide the following information - vehicle registration number, driver assigned to the vehicle, the vehicle type, the warehouse the vehicle is registered to, and the delivery hospital.

Lastly, the hospital is where deliveries are made to. Each hospital has a hospital name which identifies the hospital address, the driver identification number of the delivery vehicle assigned to make the delivery to the hospital, the hospital type, and the status of the hospital. The status can either be ‘Active’ or ‘Inactive’, which allows the system to understand whether the hospital has any pending deliveries that are due.

As a whole, the system’s primary purpose is to facilitate the management of vaccine distribution throughout a network of warehouses, delivery vehicles, and hospitals; whilst maintaining explicit constraints surrounding the database, in order to ensure data integrity is upheld.

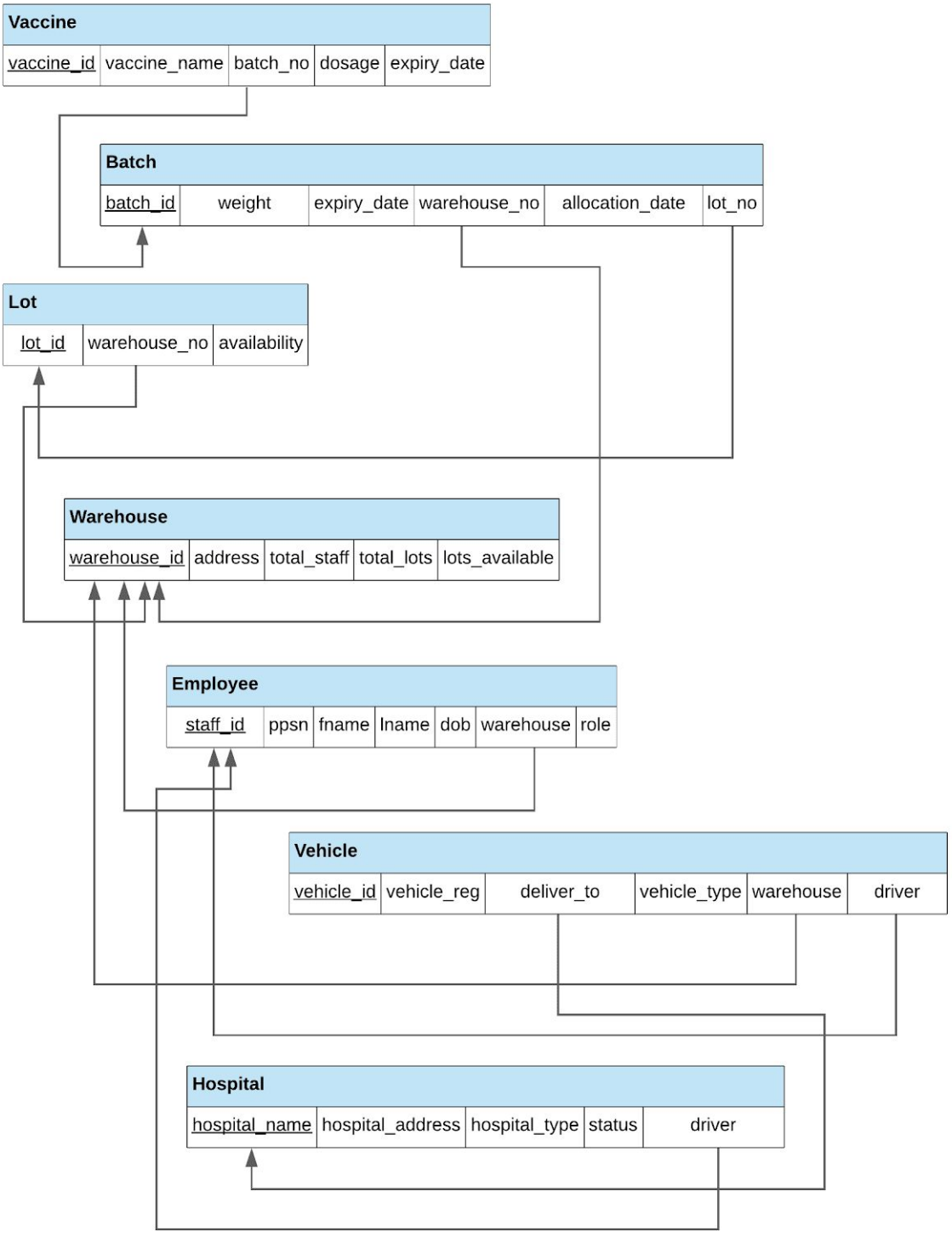
Entity Relationship Diagram

The following diagram illustrates the different tables, the attributes that they are composed of, and the relationships that exist between different tables.



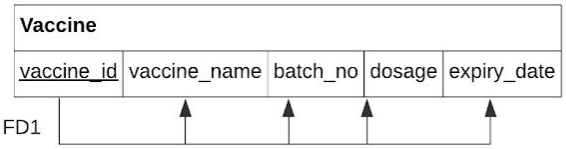
Relational Schema Diagram

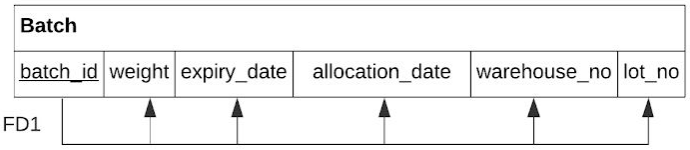
The following diagram illustrates the various relationships that attributes within each table have with other tables existing in the database.

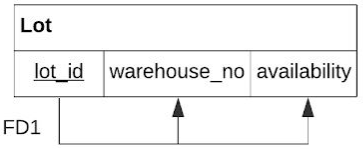


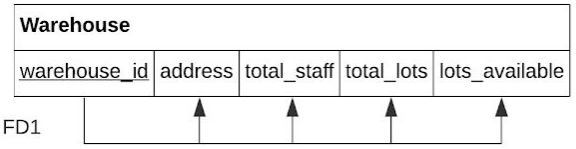
Functional Dependency Diagram

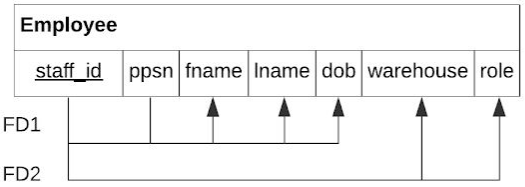
The following tables illustrate the dependencies between primary keys and other attributes within the table.

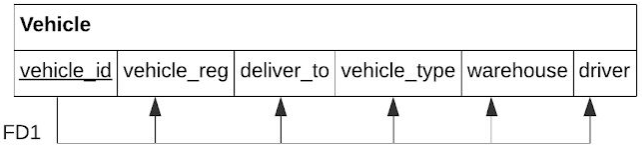


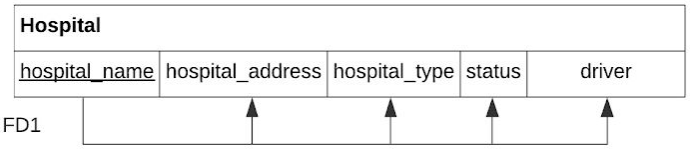












Normalisation

With this database, it was imperative that data redundancy was minimised as much as possible. The primary keys and dependencies were analysed alongside all attributes within the database to ensure all anomalies and duplicates were either removed or handled correctly.

When presented with the original tables, it was observed that the data was already compliant with 1NF. The data of all attributes could be merged and manipulated, and only a single value was permitted for each attribute. In terms of 2NF, the data did not violate the rules set out by 2NF. The schema only contained attributes which can be described by the individual table's primary key. Similarly, the data was found to be compliant with the rules set forth for 3NF as non-key attributes were not dependent on each other. Lastly, it was found that all tables were compliant with BCNF, as in 3NF the tables did not have multiple candidate keys that overlapped.

Semantic Constraints

In order to maintain data integrity throughout the database, a number of semantic constraints were applied to several tables and attributes. As a result, the risk of redundant and corrupt data is largely reduced.

From a general viewpoint - NULL values are not permitted for primary keys. The reasoning for this is to allow a unique identifier to correctly identify a specific entity.

```
lot_id INTEGER NOT NULL,
```

Additionally, a UNIQUE constraint has been implemented on some attributes, in order to prevent overlapping, and mismatches of data. An example of this would be a staff member's PPSN, as this is a government-issued identifier, and is unique to that staff member.

```
ppsn VARCHAR(20) NOT NULL UNIQUE,
```

Beginning with the vaccine table, there is a constraint on the format of the date of manufacture, as this is required by the database to verify the expiration date of the batch. The format must follow the structure 'YYYY-MM-DD'.

```
expiry_date DATE NOT NULL,
```

With hospitals, there are constraints surrounding the values that are permitted for certain fields, such as the type of hospital, and whether the hospital status is active or inactive.

```
CONSTRAINT check_hospital_status CHECK (status IN ('Active','Inactive')),
CONSTRAINT check_hospital_type CHECK (hospital_type IN ('Public','Private'))
```

With regard to lots and it's availability, there is a constraint that only permits the availability attribute value in boolean form.

```
availability BOOLEAN,
...
INSERT INTO Lot VALUES (317, 3, TRUE);
```

For an employee, there are general constraints applied to all entries. Examples include verifying that the PPSN provided is of the correct length. The Irish government issue a PPSN with 7 numbers followed by one or two letters. Additionally, the staff identification number is constrained to under the value of 1000, and the role of staff members is limited to three positions only.

```
CONSTRAINT check_ppsn_valid CHECK((LENGTH(ppsn) > 7) AND (LENGTH(ppsn) <= 9)),
CONSTRAINT check_id_valid CHECK((LENGTH(staff_id) > 0) AND (LENGTH(staff_id) <= 999)),
CONSTRAINT check_role_valid CHECK (role IN ('Driver', 'Manager', 'Operator'))
```

From the warehouse perspective, the available lots value cannot exceed that of the total lots in the warehouse, and the warehouse identification number is constrained to under 15.

```
CONSTRAINT check_lots_valid CHECK((total_lots >= lots_available)),
CONSTRAINT check_warehouse_valid CHECK(warehouse_id > 0 AND warehouse_id < 15),
```

Lastly, there are constraints on foreign keys which prevent the parent keys from being deleted if there is a child table using the parent key, and if the parent key is updated, the new value is ‘cascaded’ down to the child key to ensure data integrity.

```
FOREIGN KEY (warehouse_no) REFERENCES Warehouse(warehouse_id)
ON DELETE RESTRICT
ON UPDATE CASCADE
```

In regards to triggers within the database, there are many. Firstly, the trigger for when a new lot is added. It’s imperative that the total lots value in the relevant warehouse is incremented, as well as the availability of the lots in the warehouse. The below trigger handles this.

```
DELIMITER $$

CREATE TRIGGER addLot AFTER INSERT ON Lot
AS
BEGIN
   UPDATE Warehouse
      SET total_lots = total_lots + 1, lots_available = lots_available + 1
      WHERE warehouse_id = new.warehouse_no AND new.availability = TRUE;
END;$$

DELIMITER ;
```

When there is a new staff member, the total staff value in the relevant warehouse must be incremented to allow managers to review the capacity of the warehouse easily.

```
DELIMITER $$

CREATE TRIGGER addStaff AFTER INSERT ON Employee
AS
BEGIN
   UPDATE Warehouse
      SET total_staff = total_staff + 1
      WHERE warehouse_id = new.warehouse;
END;$$

DELIMITER ;
```

Next, the database should handle the removal of a lot by decrementing first the available lots, then the total lots from the relevant warehouse. A note is that the reason the trigger decrements the available lots before the total lots is to adhere to the constraint that prevents the available lots exceeding the total lots.

```
DELIMITER $$

CREATE TRIGGER removeLot AFTER DELETE ON Lot
AS
BEGIN
   UPDATE Warehouse
      SET lots_available = lots_available - 1, total_lots = total_lots - 1
      WHERE warehouse_id = old.warehouse;
END;$$

DELIMITER ;
```

The database shall also handle if a lot is manually updated to available. The below trigger handles this.

```
DELIMITER $$

CREATE TRIGGER increaseAvailableLots AFTER UPDATE ON Lot
AS
BEGIN
   UPDATE Warehouse
      SET lots_available = lots_available + 1
      WHERE warehouse_id = old.warehouse AND availability = TRUE;
END;$$

DELIMITER ;
```

Lastly, the database handles if a lot's availability is manually updated to unavailable. This means the lots available value in the relevant warehouse must be updated accordingly. The trigger below handles this request.

```
DELIMITER $$

CREATE TRIGGER decreaseAvailableLots AFTER UPDATE ON Lot
AS
BEGIN
   UPDATE Warehouse
      SET lots_available = lots_available - 1
      WHERE warehouse_id = old.warehouse AND availability = FALSE;
END;$$

DELIMITER ;
```

Security Policy

In terms of security, it's imperative that the right users have the correct and designated access required to the database. To handle this, roles were created for different types of users within the database. These roles were; managers, operators and drivers - each role has a specific set of permissions in terms of accessing and updating the database. Managers are permitted to create, delete and update all tables in the database, whilst operators and drivers are only permitted to update certain tables that are deemed relevant to their role.

```
CREATE ROLE Manager;
GRANT CREATE Employee TO Manager;
GRANT DELETE Employee TO Manager;
GRANT UPDATE Employee TO Manager;
...

CREATE ROLE Driver;
GRANT UPDATE Vehicle TO Driver;
...

CREATE ROLE Operator;
...

GRANT Manager TO 'Milo Greene';
...
```

Additionally, the database has different views created, which restricts the data presented - allowing managers to critically evaluate relevant data quickly. An example is the vehicles that have been assigned a driver. The "drivers_assigned_vehicles" view shows the names of the drivers, their PPSN, staff ID and the registration of the vehicle. There is an example of the code for the view, the query and the result below.

/*Create View Schema*/

```
CREATE VIEW drivers_assigned_vehicles AS
SELECT Employee.fname AS 'Driver First Name',
  Employee.lname AS 'Driver Last Name',
  Employee.ppsn AS 'Driver PPSN',
  Vehicle.driver AS 'Driver Staff Number',
  Vehicle.vehicle_reg AS 'Vehicle Registration'
FROM Vehicle
INNER JOIN Employee ON Vehicle.driver=Employee.staff_id;
```

/*SQL Query*/

```
SELECT * FROM drivers_assigned_vehicles;
```

/*SQL Result*/

Driver First Name	Driver Last Name	Driver PPSN	Driver Staff Number	Vehicle Registration
John	Smith	2846277E	212	131-C-19381
Grant	Haynes	8373826R	118	131-C-19382
Farah	Willow	2823238UE	428	131-C-19383
Sandra	Wells	2846245E	836	131-C-19384
Alexa	Hynes	8243826R	208	131-C-19385

There are more views created that show all available lots with their corresponding warehouse addresses. Additionally, in order to present the data in a visually appealing manner, the columns in the views have aliases, so instead of viewing ‘fname’, the view will see ‘Driver First Name’. By implementing these policies, it prevents data within the database being corrupted, removed accidently, or updated accidentally, which in turn preserves the integrity of the data.

Implementation

- A minimum of 6 relational tables
    - Note: Completed with more than the requirement
- Appropriate implicit constraints (including primary & foreign keys)
    - Note: Completed with more than the requirement
- Explicit (semantic) constraints such as table constraint(s) and triggers (you need not implement assertions)
    - Note: Completed with more than the requirement
- At least one view should also be defined within the database
    - Note: Completed with more than the requirement
- A minimum of 5 tuples per table
    - Note: Completed with more than the requirement
- All relevant code is attached in the appendix of this report. In addition, the ‘Schema.sql’ file is uploaded with this project report, alongside the ‘Query.sql’. Please note that the query sql file was primarily used to test only the views, hence why it is minimal. The schema file is divided into sections which has been labelled for the convenience of the reader.

Appendix

Table Creation

```
CREATE TABLE Warehouse(
warehouse_id INTEGER NOT NULL,
address VARCHAR(50) NOT NULL,
total_staff INTEGER,
total_lots INTEGER,
lots_available INTEGER,
PRIMARY KEY (warehouse_id),
CONSTRAINT check_lots_valid CHECK((total_lots >= lots_available)),
CONSTRAINT check_warehouse_valid CHECK(warehouse_id > 0 AND warehouse_id < 15)
);

CREATE TABLE Employee(
staff_id INTEGER NOT NULL,
ppsn VARCHAR(20) NOT NULL UNIQUE,
fname VARCHAR(30) NOT NULL,
lname VARCHAR(30) NOT NULL,
warehouse INTEGER NOT NULL,
role VARCHAR(50) NOT NULL,
dob DATE NOT NULL,
PRIMARY KEY (staff_id),
FOREIGN KEY (warehouse) REFERENCES Warehouse(warehouse_id)
ON DELETE RESTRICT
ON UPDATE CASCADE,
CONSTRAINT check_ppsn_valid CHECK((LENGTH(ppsn) > 7) AND (LENGTH(ppsn) <= 9)),
CONSTRAINT check_id_valid CHECK((LENGTH(staff_id) > 0) AND (LENGTH(staff_id) <= 999)),
CONSTRAINT check_role_valid CHECK (role IN ('Driver', 'Manager', 'Operator'))
);

CREATE TABLE Lot(
lot_id INTEGER NOT NULL,
warehouse_no INTEGER NOT NULL,
availability BOOLEAN,
PRIMARY KEY (lot_id),
FOREIGN KEY (warehouse_no) REFERENCES Warehouse(warehouse_id)
ON DELETE RESTRICT
ON UPDATE CASCADE
);

CREATE TABLE Batch(
batch_id INTEGER NOT NULL,
lot_no INTEGER NOT NULL,
warehouse_no INTEGER NOT NULL,
```

```
weight VARCHAR(50) NOT NULL,
expiry_date DATE NOT NULL,
allocation_date DATE NOT NULL,
PRIMARY KEY (batch_id),
FOREIGN KEY (warehouse_no) REFERENCES Warehouse(warehouse_id)
ON DELETE RESTRICT
ON UPDATE CASCADE,
FOREIGN KEY (lot_no) REFERENCES Lot(lot_id)
ON DELETE RESTRICT
ON UPDATE CASCADE
);

CREATE TABLE Vaccine(
vaccine_id INTEGER NOT NULL,
batch_no INTEGER NOT NULL,
vaccine_name VARCHAR(50) NOT NULL,
dosage VARCHAR(20) NOT NULL,
expiry_date DATE NOT NULL,
PRIMARY KEY (vaccine_id),
FOREIGN KEY (batch_no) REFERENCES Batch(batch_id)
ON DELETE RESTRICT
ON UPDATE CASCADE
);

CREATE TABLE Hospital(
hospital_name VARCHAR(50) NOT NULL,
hospital_address VARCHAR(150) NOT NULL,
hospital_type VARCHAR(30) NOT NULL,
status VARCHAR(10) NOT NULL DEFAULT 'Inactive',
driver INTEGER,
PRIMARY KEY (hospital_name),
FOREIGN KEY (driver) REFERENCES Employee(staff_id)
ON DELETE RESTRICT
ON UPDATE CASCADE,
CONSTRAINT check_hospital_status CHECK (status IN ('Active','Inactive')),
CONSTRAINT check_hospital_type CHECK (hospital_type IN ('Public','Private'))
);

CREATE TABLE Vehicle(
vehicle_id INTEGER NOT NULL,
vehicle_reg VARCHAR(50) NOT NULL,
deliver_to VARCHAR(50),
vehicle_type VARCHAR(50) NOT NULL,
warehouse INTEGER NOT NULL,
driver INTEGER NOT NULL,
```

```
PRIMARY KEY (vehicle_id),
FOREIGN KEY (driver) REFERENCES Employee(staff_id)
ON DELETE RESTRICT
ON UPDATE CASCADE,
FOREIGN KEY (warehouse) REFERENCES Warehouse(warehouse_id)
ON DELETE RESTRICT
ON UPDATE CASCADE
);
```

Triggers

```
DELIMITER $$
CREATE TRIGGER addLot AFTER INSERT ON Lot
AS
BEGIN
   UPDATE Warehouse
      SET total_lots = total_lots + 1, lots_available = lots_available + 1
      WHERE warehouse_id = new.warehouse_no AND new.availability = TRUE;
END;$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER addStaff AFTER INSERT ON Employee
AS
BEGIN
   UPDATE Warehouse
      SET total_staff = total_staff + 1
      WHERE warehouse_id = new.warehouse;
END;$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER removeLot AFTER DELETE ON Lot
AS
BEGIN
   UPDATE Warehouse
      SET lots_available = lots_available - 1, total_lots = total_lots - 1
      WHERE warehouse_id = old.warehouse;
END;$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER increaseAvailableLots AFTER UPDATE ON Lot
```

```
AS
BEGIN
   UPDATE Warehouse
      SET lots_available = lots_available + 1
      WHERE warehouse_id = old.warehouse AND availability = TRUE;
END;$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER decreaseAvailableLots AFTER UPDATE ON Lot
AS
BEGIN
   UPDATE Warehouse
      SET lots_available = lots_available - 1
      WHERE warehouse_id = old.warehouse AND availability = FALSE;
END;$$
DELIMITER ;
```

Database Population

```
INSERT INTO Warehouse VALUES (1, 'Cork, Ireland', 0, 0, 0);
INSERT INTO Warehouse VALUES (2, 'Dublin, Ireland', 0, 0, 0);
INSERT INTO Warehouse VALUES (3, 'Galway, Ireland', 0, 0, 0);
INSERT INTO Warehouse VALUES (4, 'Limerick, Ireland', 0, 0, 0);
INSERT INTO Warehouse VALUES (5, 'Sligo, Ireland', 0, 0, 0);
INSERT INTO Warehouse VALUES (6, 'Wexford, Ireland', 0, 0, 0);

INSERT INTO Employee VALUES (212, '2846277E', 'John', 'Smith', 2, 'Driver', '1982-09-23' );
INSERT INTO Employee VALUES (238, '8295625E', 'Mary', 'Adams', 5, 'Operator', '1993-02-01' );
INSERT INTO Employee VALUES (844, '2810386YF', 'Milo', 'Greene', 3, 'Manager', '1976-07-10' );
INSERT INTO Employee VALUES (118, '8373826R', 'Grant', 'Haynes', 2, 'Driver', '1991-12-08' );
INSERT INTO Employee VALUES (428, '2823238UE', 'Farah', 'Willow', 4, 'Driver', '1962-03-25' );
INSERT INTO Employee VALUES (248, '2840719D', 'Sarah', 'Davis', 3, 'Operator', '1995-11-21' );
INSERT INTO Employee VALUES (836, '2846245E', 'Sandra', 'Wells', 3, 'Driver', '1998-01-23' );
INSERT INTO Employee VALUES (318, '8295324E', 'Jamie', 'Stock', 5, 'Operator', '1978-02-01' );
INSERT INTO Employee VALUES (102, '2810323YF', 'Richard', 'Fitzpatrick', 3, 'Manager', '1986-07-10' );
INSERT INTO Employee VALUES (208, '8243826R', 'Alexa', 'Hynes', 2, 'Driver', '1997-12-08' );
INSERT INTO Employee VALUES (142, '2842232FE', 'William', 'Skerry', 4, 'Driver', '1968-03-25' );
INSERT INTO Employee VALUES (104, '2840721R', 'Ciara', 'Murray', 6, 'Operator', '1991-11-21' );

INSERT INTO Lot VALUES (317, 3, TRUE);
INSERT INTO Lot VALUES (333, 3, TRUE);
INSERT INTO Lot VALUES (113, 1, TRUE);
```

```
INSERT INTO Lot VALUES (414, 4, TRUE);
INSERT INTO Lot VALUES (294, 2, TRUE);
INSERT INTO Lot VALUES (319, 3, TRUE);
INSERT INTO Lot VALUES (518, 5, TRUE);
INSERT INTO Lot VALUES (495, 4, TRUE);
INSERT INTO Lot VALUES (228, 2, TRUE);
INSERT INTO Lot VALUES (195, 1, TRUE);
INSERT INTO Lot VALUES (557, 5, TRUE);
INSERT INTO Lot VALUES (528, 5, TRUE);
INSERT INTO Lot VALUES (174, 2, TRUE);
INSERT INTO Lot VALUES (418, 4, TRUE);
INSERT INTO Lot VALUES (559, 5, TRUE);
INSERT INTO Lot VALUES (853, 3, TRUE);
INSERT INTO Lot VALUES (219, 2, TRUE);
INSERT INTO Lot VALUES (295, 2, TRUE);
INSERT INTO Lot VALUES (618, 6, TRUE);
INSERT INTO Lot VALUES (385, 3, TRUE);

INSERT INTO Batch VALUES (242, 317, 3, '30438g', '2020-12-30', '2020-12-01');
INSERT INTO Batch VALUES (424, 333, 3, '30438g', '2020-12-30', '2020-12-01');
INSERT INTO Batch VALUES (145, 113, 1, '30438g', '2020-12-31', '2020-12-02');
INSERT INTO Batch VALUES (732, 414, 4, '30438g', '2020-12-31', '2020-12-02');
INSERT INTO Batch VALUES (827, 294, 2, '30438g', '2020-12-31', '2020-12-02');
INSERT INTO Batch VALUES (917, 319, 3, '30438g', '2020-12-31', '2020-12-02');
INSERT INTO Batch VALUES (149, 518, 5, '30438g', '2020-12-31', '2020-12-02');
INSERT INTO Batch VALUES (184, 495, 4, '30438g', '2020-12-31', '2020-12-02');
INSERT INTO Batch VALUES (194, 228, 2, '30438g', '2020-12-31', '2020-12-02');
INSERT INTO Batch VALUES (645, 195, 1, '30438g', '2020-12-31', '2020-12-02');
INSERT INTO Batch VALUES (143, 557, 5, '30438g', '2020-12-31', '2020-12-02');
INSERT INTO Batch VALUES (329, 528, 5, '30438g', '2020-12-31', '2020-12-02');

INSERT INTO Vaccine VALUES (2834, 242, 'CoronaVac Sinovac', '1', '2020-12-30');
INSERT INTO Vaccine VALUES (2835, 242, 'CoronaVac Sinovac', '1', '2020-12-30');
INSERT INTO Vaccine VALUES (2836, 242, 'CoronaVac Sinovac', '1', '2020-12-30');
INSERT INTO Vaccine VALUES (2837, 242, 'CoronaVac Sinovac', '1', '2020-12-30');
INSERT INTO Vaccine VALUES (2838, 242, 'CoronaVac Sinovac', '1', '2020-12-30');
INSERT INTO Vaccine VALUES (2839, 242, 'CoronaVac Sinovac', '1', '2020-12-30');
INSERT INTO Vaccine VALUES (2840, 242, 'CoronaVac Sinovac', '1', '2020-12-30');

INSERT INTO Vaccine VALUES (2934, 145, 'CoronaVac Sinovac', '1', '2020-12-31');
INSERT INTO Vaccine VALUES (2935, 145, 'CoronaVac Sinovac', '1', '2020-12-31');
INSERT INTO Vaccine VALUES (2936, 145, 'CoronaVac Sinovac', '1', '2020-12-31');
INSERT INTO Vaccine VALUES (2937, 145, 'CoronaVac Sinovac', '1', '2020-12-31');
INSERT INTO Vaccine VALUES (2938, 145, 'CoronaVac Sinovac', '1', '2020-12-31');
INSERT INTO Vaccine VALUES (2939, 145, 'CoronaVac Sinovac', '1', '2020-12-31');
```

```
INSERT INTO Vaccine VALUES (2940, 145, 'CoronaVac Sinovac', '1', '2020-12-31');

INSERT INTO Vaccine VALUES (3834, 827, 'CoronaVac Sinovac', '1', '2020-12-31');
INSERT INTO Vaccine VALUES (3835, 827, 'CoronaVac Sinovac', '1', '2020-12-31');
INSERT INTO Vaccine VALUES (3836, 827, 'CoronaVac Sinovac', '1', '2020-12-31');
INSERT INTO Vaccine VALUES (3837, 827, 'CoronaVac Sinovac', '1', '2020-12-31');
INSERT INTO Vaccine VALUES (3838, 827, 'CoronaVac Sinovac', '1', '2020-12-31');
INSERT INTO Vaccine VALUES (3839, 827, 'CoronaVac Sinovac', '1', '2020-12-31');
INSERT INTO Vaccine VALUES (3840, 827, 'CoronaVac Sinovac', '1', '2020-12-31');

INSERT INTO Vaccine VALUES (2834, 194, 'CoronaVac Sinovac', '1', '2020-12-31');
INSERT INTO Vaccine VALUES (2835, 194, 'CoronaVac Sinovac', '1', '2020-12-31');
INSERT INTO Vaccine VALUES (2836, 194, 'CoronaVac Sinovac', '1', '2020-12-31');
INSERT INTO Vaccine VALUES (2837, 194, 'CoronaVac Sinovac', '1', '2020-12-31');
INSERT INTO Vaccine VALUES (2838, 194, 'CoronaVac Sinovac', '1', '2020-12-31');
INSERT INTO Vaccine VALUES (2839, 194, 'CoronaVac Sinovac', '1', '2020-12-31');
INSERT INTO Vaccine VALUES (2840, 194, 'CoronaVac Sinovac', '1', '2020-12-31');

INSERT INTO Hospital VALUES ('Mercy', 'St. Patricks Street, Cork', 'Private', 'Active', 212);
INSERT INTO Hospital VALUES ('St. James', 'The Liberties, Dublin 8', 'Public', 'Active', 118);
INSERT INTO Hospital VALUES ('UCD', 'Donnybrook, Dublin', 'Public', 'Active', 428);
INSERT INTO Hospital VALUES ('Bonds', 'Sundays Wells, Cork', 'Private', 'Active', 836);
INSERT INTO Hospital VALUES ('Kinsale Hospital', 'Main Street, Cork', 'Public', 'Active', 208);

INSERT INTO Vehicle VALUES (12, '131-C-19381', 'Mercy', 'Temperature-Controlled Van', 2, 212);
INSERT INTO Vehicle VALUES (13, '131-C-19382', 'St. James', 'Temperature-Controlled Van', 2, 118);
INSERT INTO Vehicle VALUES (14, '131-C-19383', 'UCD', 'Temperature-Controlled Van', 4, 428);
INSERT INTO Vehicle VALUES (15, '131-C-19384', 'Bonds', 'Temperature-Controlled Van', 3, 836);
INSERT INTO Vehicle VALUES (16, '131-C-19385', 'Kinsale Hospital', 'Temperature-Controlled Van', 2, 208);
```

Role Creation

```
CREATE ROLE Manager;
GRANT CREATE Employee TO Manager;
GRANT DELETE Employee TO Manager;
GRANT UPDATE Employee TO Manager;
GRANT CREATE Hospital TO Manager;
GRANT DELETE Hospital TO Manager;
GRANT UPDATE Hospital TO Manager;
GRANT CREATE Vehicle TO Manager;
GRANT DELETE Vehicle TO Manager;
GRANT UPDATE Vehicle TO Manager;
GRANT CREATE Batch TO Manager;
```

```
GRANT DELETE Batch TO Manager;
GRANT UPDATE Batch TO Manager;
GRANT CREATE Lot TO Manager;
GRANT DELETE Lot TO Manager;
GRANT UPDATE Lot TO Manager;
GRANT CREATE Vaccine TO Manager;
GRANT DELETE Vaccine TO Manager;
GRANT UPDATE Vaccine TO Manager;
GRANT CREATE Warehouse TO Manager;
GRANT DELETE Warehouse TO Manager;
GRANT UPDATE Warehouse TO Manager;

CREATE ROLE Driver;
GRANT UPDATE Vehicle TO Driver;
GRANT UPDATE Hospital TO Driver;

CREATE ROLE Operator;
GRANT UPDATE Warehouse TO Operator;
GRANT UPDATE Batch TO Operator;
GRANT UPDATE Lot TO Operator;

GRANT Manager TO 'Milo Greene';
GRANT Driver TO 'John Smith';
GRANT Operator TO 'Sarah Davis';
```

View Creation

```
CREATE VIEW drivers_assigned_vehicles AS
SELECT Employee.fname AS 'Driver First Name',
  Employee.lname AS 'Driver Last Name',
  Employee.ppsn AS 'Driver PPSN',
  Vehicle.driver AS 'Driver Staff Number'
FROM Vehicle
INNER JOIN Employee ON Vehicle.driver=Employee.staff_id;

CREATE VIEW lot_locations AS
SELECT Warehouse.warehouse_id AS 'Warehouse',
  Warehouse.address AS 'Address',
  Lot.lot_id AS 'Location (Lot)'
FROM Warehouse
INNER JOIN Lot
ON Warehouse.warehouse_id = Lot.warehouse_no AND Lot.availability = TRUE;
```