**Tasks Week 1: Database Normalization and Export Guide**

**1. Introduction**

This document serves as a step-by-step guide to database normalization (1NF → 2NF → 3NF) and exporting the database from SQLiteStudio.

**2. Database Normalization Steps**

**Step 1: Convert to First Normal Form (1NF)**

* The initial table contained **repeating groups** in the ProductName and ItemsOrdered columns.
* To fix this, each product was placed in a **separate row**.
* The modified table in **1NF** ensured that each column held **atomic values**.

✅ **SQL Commands Used:**

CREATE TABLE Orders (

OrderID INTEGER PRIMARY KEY AUTOINCREMENT,

CustomerName TEXT NOT NULL,

PostCode TEXT NOT NULL,

Phone TEXT NOT NULL,

StockName TEXT NOT NULL,

ProductName TEXT NOT NULL,

ItemsOrdered INTEGER NOT NULL,

ItemsLeftInStock INTEGER NOT NULL

);

✅ **Inserted Data** (Sample):

INSERT INTO Orders (CustomerName, PostCode, Phone, StockName, ProductName, ItemsOrdered, ItemsLeftInStock)

VALUES

('Jake', 'SO155UP', '0775252362', 'Southampton', 'Pomegranate', 1, 99),

('Jake', 'SO155UP', '0775252362', 'Southampton', 'Lime', 2, 18);

**Step 2: Convert to Second Normal Form (2NF)**

* **Issue in 1NF:** Redundant customer details (name, postcode, phone) and stock information.
* **Solution:** Split into **three separate tables**: Customers, Products, and Orders.

✅ **Created Customers Table:**

CREATE TABLE Customers (

CustomerID INTEGER PRIMARY KEY AUTOINCREMENT,

CustomerName TEXT NOT NULL,

PostCode TEXT NOT NULL,

Phone TEXT NOT NULL

);

✅ **Created Products Table:**

CREATE TABLE Products (

ProductID INTEGER PRIMARY KEY AUTOINCREMENT,

ProductName TEXT NOT NULL,

StockName TEXT NOT NULL

);

✅ **Updated Orders Table:**

CREATE TABLE Orders (

OrderID INTEGER PRIMARY KEY AUTOINCREMENT,

CustomerID INTEGER NOT NULL,

ProductID INTEGER NOT NULL,

ItemsOrdered INTEGER NOT NULL,

ItemsLeftInStock INTEGER NOT NULL,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

✅ **Inserted Data into Customers & Products:**

INSERT INTO Customers (CustomerName, PostCode, Phone)

VALUES

('Jake', 'SO155UP', '0775252362'),

('James', 'SO169OP', '0775523256');

**Step 3: Convert to Third Normal Form (3NF)**

* **Issue in 2NF:** StockName depends on ProductName, causing a **transitive dependency**.
* **Solution:** Move StockName to a separate **Stock Table**.

✅ **Created Stock Table:**

CREATE TABLE Stock (

StockID INTEGER PRIMARY KEY AUTOINCREMENT,

StockName TEXT NOT NULL UNIQUE

);

✅ **Updated Products Table:**

CREATE TABLE Products (

ProductID INTEGER PRIMARY KEY AUTOINCREMENT,

ProductName TEXT NOT NULL,

StockID INTEGER NOT NULL,

FOREIGN KEY (StockID) REFERENCES Stock(StockID)

);

✅ **Final Verification:**

SELECT \* FROM Customers;

SELECT \* FROM Products;

SELECT \* FROM Stock;

SELECT \* FROM Orders;

**3. Exporting the Database**

**Option 1: Locate and Copy .db File**

1. Open **SQLiteStudio**.
2. Click **Database → Database List**.
3. Find OrdersDB and note the file path.
4. Open **File Explorer** and navigate to that location.
5. **Copy** OrdersDB.sqlite (or .db) file to a safe location.

**Option 2: Export SQL File**

1. Right-click on OrdersDB in SQLiteStudio.
2. Click **Export Database** → **SQL File**.
3. Select **all tables** and choose a save location.
4. Click **OK** to save as OrdersDB\_Export.sql.

**4. Conclusion**

This document outlines the **step-by-step process** of normalizing a database and exporting it for backup or submission. Following this structure ensures an **efficient, well-structured, and optimized database**.

✅ **Normalization Complete (1NF → 2NF → 3NF)** ✅ **Database Successfully Exported**