

Table of Contents

Task 1.....	2
1. Introduction.....	2
1. Logical Model	2
1.1. Entities for Parkside Medical practice	2
2.2 Attributes for each entity	2
2.3 Relationships.....	4
2.4. Logical Model (Entity Relationship Diagram).....	4
2.5. Normalization of 3 NF	5
2.6. Assumptions.....	6
2. Physical Model	7
2.1. Entities	7
2.2. Changes made to logical model:	8
Impact of changes made from logical to physical model.....	8
2.3. Physical Model (Entity Relationship Diagram).....	9
Task 2.....	10
1. Introduction	10
2. Screenshots of Tables	10
2.1. Staff Table	10
2.2. Supplier Table	11
2.3. Product Table	12
2.4. PurchaseOrder Table.....	13
2.5. PurchaseOrderLine Table	14
2.6. Delivery Table.....	15
2.7. Payment Table	16
Task3.....	17
Introduction	17
Case 1:	17
Case 2:	18
Case 3	20
Case 4	25
References	28

Task 1

1. Introduction

Parkside Medical Practice is a surgery that employs general practitioners, nurses and administrative staff. They provide healthcare to a variety of patients with different medical needs. This report shows the design process for a database that will help manage the surgery's operations more effectively. It includes identifying entities, attributes and defining the relationships between them.

1. Logical Model

1.1. Entities for Parkside Medical practice

1. **Patient:** Store patient information.
2. **Appointment:** Store appointment details.
3. **Prescription:** Store prescription information.
4. **Staff:** Store staff details.
5. **Role:** Store staff roles.

2.2 Attributes for each entity

1. Patient:

- o Patient_id (PK)
- o First_Name
- o Last_Name
- o Phone_Number
- o Email_Address
- o DOB
- o Registration_Date

2. Appointment:

- Appointment_id (PK)
- Patient_id (FK)
- Staff_id (FK)
- Appointment_Type
- Appointment_Status
- Date
- Time
- Treatment_Notes
- Diagnosis

3. Prescription:

- Prescription_id (PK)
- Appointment_id (FK)
- Medication
- Quantity
- Prescription_Instructions
- Date_Issued

4. Staff:

- Staff_id (PK)
- Role_id (FK)
- First_Name
- Last_Name
- Date_Joined

5. Role:

- Role_id (PK)
- Rol_Name

2.3 Relationships

- Patients may have zero or many appointments.
- Each appointment must have exactly one patient.
- A Staff members may have zero or many appointments.
- Exactly one staff member may conduct an appointment.
- Each appointment may issue zero or many prescriptions.
- Each prescription is linked to exactly one appointment.
- Each Staff member must have exactly one role.
- A Roles can be assigned to one or many staff members.

2.4. Logical Model (Entity Relationship Diagram)

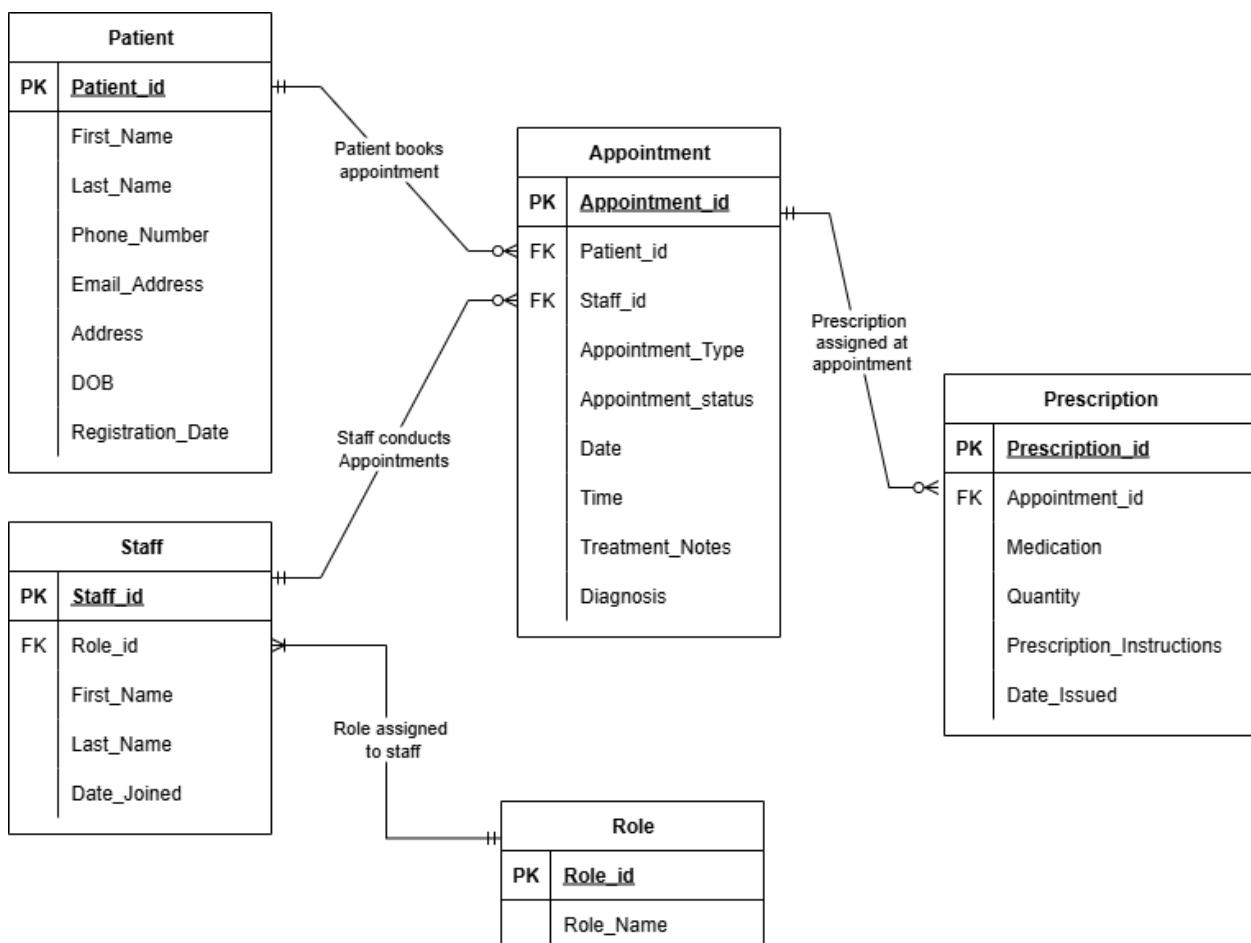


Figure 1 - Logical Model

2.5. Normalization of 3 NF

Each entity was normalized to 3NF, this ensures that the relationships obtained from data models are free from redundancy which can bring about update, insert and delete inconsistency when applied (Connolly & Begg, 2015).

1. Patient:

- **1NF:**

Repeating data was removed.

- **2NF:**

Each attribute depends fully on a single key, Patient_id.

- **3NF:**

There are no transitive dependencies.

2. Appointment:

- **1NF:**

Appointment details were separated from patient and staff information.

- **2NF:**

A single key (Appointment_id) was created so that each attribute only depends on one key.

- **3NF:**

There are no transitive dependencies.

3. Prescription:

- **1NF:**

Prescription details were separated from appointment information.

- **2NF:**

Each attribute depends fully on a single key, Prescription_id.

- **3NF:**

There are no transitive dependencies.

4. Staff:

- **1NF:**

Staff details were separated from their roles.

- **2NF:**

Each attribute depends fully on a single key, Staff_id.

- **3NF:**

A transitive dependency between Staff and Role was removed by creating a separate Role entity.

5. **Role:**

- **1NF:**

Role details have only atomic values.

- **2NF:**

All attributes depend on a single key, Role_id.

- **3NF:**

There are no transitive dependencies.

2.6. **Assumptions**

- Only registered patients can book appointments, this ensures that every appointment is linked to a patient record.
- Appointments can only have a single patient at a time, no group bookings.
- Only GPs and nurses can conduct appointments.
- Administrative staff may not conduct appointments, this prevents non-medical staff from conducting appointments.
- Prescriptions are linked to appointments, not directly to patients or staff, this avoids data redundancy, as details of the prescription, staff member and patient can be obtained through appointments.

2. Physical Model

2.1. Entities

1. Patient:

- Patient_id INT (PK, AUTO_INCREMENT)
- First_Name VARCHAR(50) NOT NULL
- Last_Name VARCHAR(50) NOT NULL
- Phone_Number VARCHAR(20) NOT NULL
- Email_Address VARCHAR(100) UNIQUE
- DOB DATE NOT NULL
- Registration_Date DATE NOT NULL

2. Appointment:

- Appointment_id INT (PK, AUTO_INCREMENT)
- Patient_id (FK) INT NOT NULL
- Staff_id (FK) INT NOT NULL
- Appointment_Type VARCHAR(50) NOT NULL
- Appointment_Status VARCHAR(50) NOT NULL
- Date DATE NOT NULL
- Time TIME NOT NULL
- Treatment_Notes TEXT
- Diagnosis VARCHAR(250)

3. Prescription:

- Prescription_id INT (PK, AUTO_INCREMENT)
- Appointment_id INT (FK) NOT NULL
- Medication VARCAHR9100) NOT NULL
- Quantity INT NOT NULL
- Prescription_Instructions TEXT
- Date_Issued DATE NOT NULL

4. Staff:

- Staff_id INT (PK, AUTO_INCREMENT)
- Role_id INT (FK) NOT NULL
- First_Name VARCHAR(50) NOT NULL
- Last_Name VARCHAR(50) NOT NULL
- Date_Joined DATE NOT NULL

5. Role:

- Role_id INT (PK, AUTO_INCREMENT)
- Rol_Name VARCHAR(50) NOT NULL UNIQUE

2.2. Changes made to logical model:

• Primary Keys and Foreign Keys:

Primary keys were added to uniquely identify each record in these tables while foreign keys were added to link related entities to each other. (Pajankar, 2020).

• Data types and Constraints:

- VARCHAR(n):
Used for short text such as names, surnames and phone numbers.
- TEXT:
Used for longer text such as descriptions or notes.
- DATE:
Used to store dates such as Date_Issued and Date_joined.
- TIME:
Store specific appointment times.
- INT:
Used for numerical values such as ID's and quantities.
- UNIQUE:
Prevents duplicate entries, making each one of a kind.
- NULL:
Ensure that important fields are filled in.

Impact of changes made from logical to physical model

• Data types:

They were implemented into the physical model so we could identify storage methods for the data, allowing us to improve efficiency for our database (Connolly & Begg, 2015).

• Constraints:

Constraint provides data integrity and accuracy for all tables, these constraints include NOT NULL, UNIQUE and FOREIGN KEY

2.3. Physical Model (Entity Relationship Diagram)

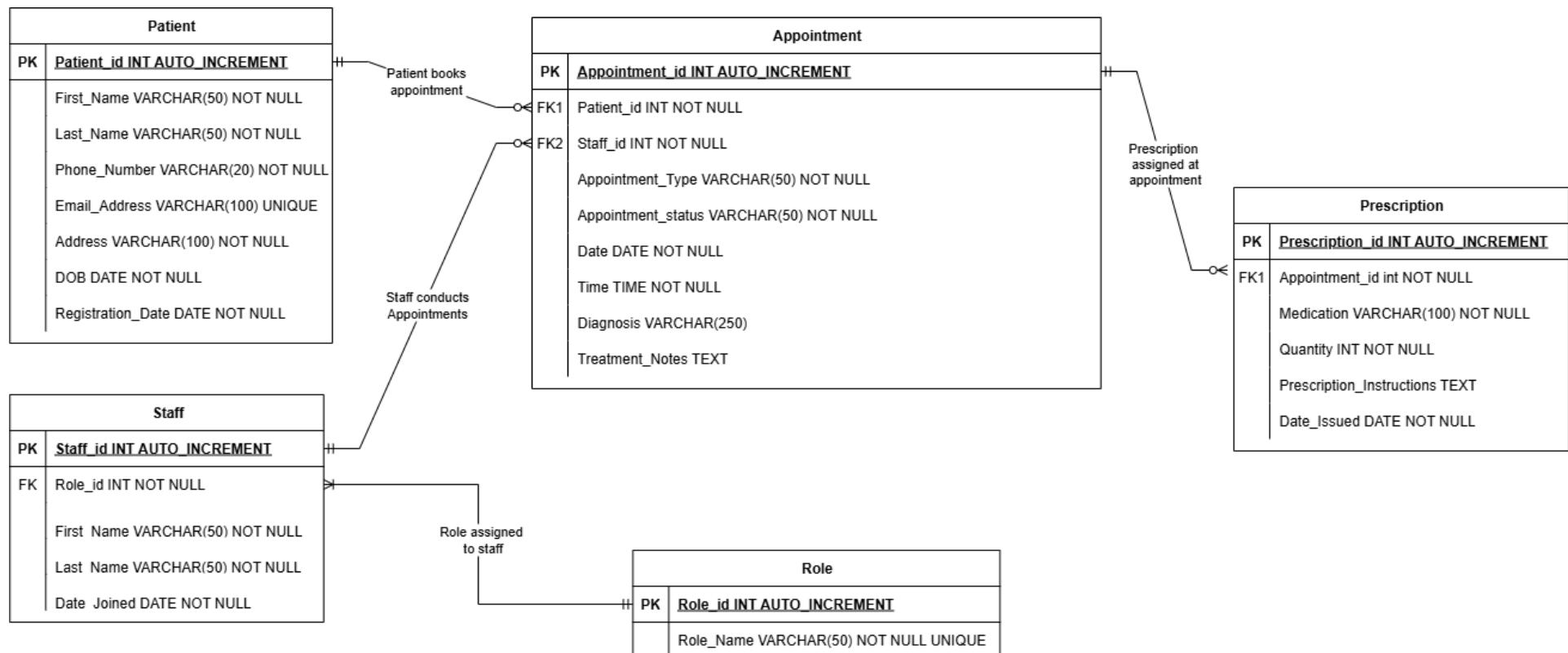


Figure 2 - Physical Model

Task 2

1. Introduction

The database was created for Parkside purchase ordering system. Seven tables were constructed using the CREATE TABLE statements with the suitable data types, primary keys, foreign keys and constraints. Auto Increment was applied where indicated to generate unique identifiers automatically.

After tables were created, relevant data was inserted into each table to show realistic records for staff, supplier, products, purchase order, purchase order line, deliveries and payments. The database was then tested to ensure relationships and constraints work as intended.

Student number was used as database name, as seen on screenshots and the full SQL code used for table creation and data insertion can be found in **Appendix 1**

2. Screenshots of Tables

2.1. Staff Table

The screenshot shows the phpMyAdmin interface for the 'Staff' table in the 'c8xpgso_24120579' database. The left sidebar shows other tables like New, Delivery, Payment, Product, PurchaseOrder, PurchaseOrderLine, Supplier, and information_schema. The main area displays the results of the query 'SELECT * FROM Staff;'. The results are:

StaffID	StaffFirstName	StaffSurname
1000	Alice	Miller
1001	John	Smith
1002	Sarah	Brown
1003	Michael	Davis
1004	Emily	Johnson
1005	Robert	Wilson

Below the table, there are buttons for Edit, Copy, Delete, and other operations. At the bottom, there are links for Print, Copy to clipboard, Export, Display chart, and Create view.

Figure 3 - Staff table

```
SELECT * FROM Staff;
```

Figure 3 shows the Staff table, which stores all staff details including StaffID, StaffFirstName and StaffSurname.

It contains six staff members each with a unique id, starting from 1000 to 1005. These IDs are used to link staff members to purchase orders, deliveries and payments.

2.2. Supplier Table

The screenshot shows the phpMyAdmin interface for the 'Supplier' table. The left sidebar lists databases and tables, with 'Supplier' selected. The main area displays the table structure and data. The table has columns: SupplierID, Name, Phone, ContactName, Email, and Address. Six rows of supplier data are listed, each with edit, copy, and delete options. The data is as follows:

SupplierID	Name	Phone	ContactName	Email	Address
100	Medicare	07760877635	Oliver Queen	oliver@medicare.com	44 Church Lane, Brighton
101	Discem	07704335370	Grant Gustin	grant@discem.com	77 Richmond Road, South West London
102	Vitality care	07019811309	Walter Black	walter@vitalitycare.com	12 Victoria Road, East London
103	Mediworld	07856473374	Gavin Booth	gavin@mediworld	838 Kingsway, Hemel Hempstead
104	MK Medicals	07859634762	Russell Roman	russell@mkmédicals.com	59 George Street, Darlington
105	Excel Healthcare	07702214012	David Kirby	david@excelhealthcare	99 Station Road, Glasgow

Figure 4 -Supplier Table

```
SELECT * FROM Supplier;
```

Figure 4 shows the Supplier table, which stores supplier details including SupplierID, Name, Phone, ContactName, Email and Address.

It contains six suppliers, each with their own unique ID from 100 to 105. These ID links them to their products and purchase orders.

2.3. Product Table

The screenshot shows the phpMyAdmin interface for a database named 'c8xpgso_24120579'. The left sidebar lists schemas: 'c8xpgso_24120579' (containing tables like New, Delivery, Payment, Product, PurchaseOrder, PurchaseOrderLine, Staff, Supplier) and 'information_schema', 'performance_schema'. The main area is titled 'Table: Product' with a sub-header 'Showing rows 0 - 18 (19 total, Query took 0.0003 seconds.)'. It displays a list of 19 products with columns: ProductID, ProductName, Description, UnitPrice, QuantityInStock, and SupplierID. Each row has edit, copy, and delete options. The data includes items like Disposable Syringes, Examination Gloves, Stethoscope, Digital Thermometer, Alcohol Swabs, Blood Pressure Monitor, Bandages, Hand Sanitizer, Surgical Masks, Office Printer Paper, Disinfectant Spray, Waiting Room Chairs, Sharps Bin, Tongue Depressors, Glucose Testing Strips, Clipboard, Wall Clock, Otoscope Set, and Floor Cleaner.

	ProductID	ProductName	Description	UnitPrice	QuantityInStock	SupplierID
<input type="checkbox"/>	500	Disposable Syringes	Sterile 5ml syringes, box of 100	15.99	80	100
<input type="checkbox"/>	501	Examination Gloves	Nitrile gloves, medium, box of 100	12.50	12	100
<input type="checkbox"/>	502	Stethoscope	Professional grade dual-head	59.99	2	100
<input type="checkbox"/>	503	Digital Thermometer	Fast-read digital thermometer	9.99	5	105
<input type="checkbox"/>	504	Alcohol Swabs	Isopropyl alcohol prep pads, 200 pcs	6.95	30	102
<input type="checkbox"/>	505	Blood Pressure Monitor	Upper arm automatic unit	75.00	12	101
<input type="checkbox"/>	506	Bandages	Self-adhesive 4-inch roll bandages	4.99	58	101
<input type="checkbox"/>	507	Hand Sanitizer	500ml antibacterial gel bottles	3.50	60	101
<input type="checkbox"/>	508	Surgical Masks	3-ply disposable face masks, 50 pack	8.20	88	102
<input type="checkbox"/>	509	Office Printer Paper	A4 80gsm white, ream of 500 sheets	4.25	10	105
<input type="checkbox"/>	510	Disinfectant Spray	Surface cleaner, 750ml bottle	2.85	12	104
<input type="checkbox"/>	511	Waiting Room Chairs	Vinyl padded stackable chairs	45.00	24	103
<input type="checkbox"/>	512	Sharps Bin	5-litre yellow container for needles	7.95	45	102
<input type="checkbox"/>	513	Tongue Depressors	Wooden, non-sterile, box of 100	3.10	70	104
<input type="checkbox"/>	514	Glucose Testing Strips	50-count, for use with glucometer	14.50	100	103
<input type="checkbox"/>	515	Clipboard	A4 plastic with metal clip	2.20	12	105
<input type="checkbox"/>	516	Wall Clock	Silent, analog, batterypowered	11.75	2	103
<input type="checkbox"/>	517	Otoscope Set	Diagnostic otoscope with 3 specula	89.99	5	103
<input type="checkbox"/>	518	Floor Cleaner	5-litre hospital-grade concentrate	16.80	34	104

Figure 5 - Product Table

```
SELECT * FROM Product;
```

Figure 5 shows the Product table, which stores product details, including ProductID, ProductName, Description, UnitPrice, QuantityInStock and SupplierID.

It contains 19 products, each with their own unique ID from 500 to 518. SupplierID is a Foreign Key that links each product to its supplier in the supplier table.

2.4. PurchaseOrder Table

The screenshot shows the phpMyAdmin interface for the 'PurchaseOrder' table. The left sidebar lists databases and tables, with 'PurchaseOrder' selected. The main area displays the table structure and data. The table has columns: PurchaseOrderID, OrderDate, StaffID, and SupplierID. Six rows of data are shown, each with unique IDs and dates ranging from January 10, 2025, to March 12, 2025.

PurchaseOrderID	OrderDate	StaffID	SupplierID
1	2025-01-10	1000	100
2	2025-01-13	1001	101
3	2025-01-15	1002	102
4	2025-02-10	1003	103
5	2025-03-01	1004	104
6	2025-03-12	1005	105

Figure 6 - PurchaseOrder Table

```
SELECT * FROM PurchaseOrder;
```

Figure 6 shows the PurchaseOrder table, which stores order details including PurchaseOrderID, OrderDate, StaffID and SupplierID.

It contains six orders, each with their own unique ID from 1 to 6.

StaffID and SupplierID are foreign keys that link each order to the staff member who placed it and the supplier it was made with.

2.5. PurchaseOrderLine Table

The screenshot shows the phpMyAdmin interface for the 'PurchaseOrderLine' table. The left sidebar lists databases and tables, with 'PurchaseOrderLine' selected. The main area displays 16 rows of data with columns: POLineID, PurchaseOrderID, ProductID, and Quantity. The data is as follows:

POLineID	PurchaseOrderID	ProductID	Quantity
1	1	500	10
2	1	501	5
3	2	505	2
4	2	507	20
5	2	506	8
6	3	508	10
7	3	512	3
8	4	511	4
9	4	516	1
10	4	517	2
11	4	514	25
12	5	510	5
13	5	518	1
14	6	503	4
15	6	509	2
16	6	515	8

Figure 7 - PurchaseOrderLine Table

```
SELECT * FROM PurchaseOrderLine;
```

Figure 6 shows the PurchaseOrderLine table, which stores the quantity of each product in a single purchase order.

It contains details such as POLineID, PurchaseOrderID, ProductID and Quantity.

PurchaseOrderID and Product ID are foreign keys that link each record to the related product and purchase order.

2.6. Delivery Table

The screenshot shows the phpMyAdmin interface for the 'Delivery' table. The left sidebar lists databases (c8xpgso_24120579) and tables (New, Delivery, Payment, Product, PurchaseOrder, PurchaseOrderLine, Staff, Supplier). The main area shows the results of the query 'SELECT * FROM Delivery;'. The results table has columns: DeliveryID, PurchaseOrderID, DeliveryDate, StaffID, and DeliveryStatus. Six rows are listed, each with edit, copy, and delete options. The data is as follows:

DeliveryID	PurchaseOrderID	DeliveryDate	StaffID	DeliveryStatus
1	1	2025-01-12	1000	Complete
2	2	2025-01-16	1001	Complete
3	3	2025-01-25	1002	Complete
4	4	2025-02-12	1003	Partial
5	5	2025-03-10	1004	Complete
6	6	2025-03-14	1005	Returned

Figure 8 - Delivery Table

```
SELECT * FROM Delivery;
```

Figure 8 shows the Delivery table, which stores details of deliveries, including DeliveryID, PurchaseOrderID, DeliveryDate, StaffID and DeliveryStatus.

It contains six deliveries, each with their own unique ID from 1 to 6. PurchaseOrderID and StaffID are foreign keys that link each delivery to the purchase order it belongs to and the staff member who handled it.

Delivery Status shows whether the delivery is completed, partial or returned.

2.7. Payment Table

The screenshot shows the phpMyAdmin interface for the 'Payment' table in the 'c8xpgso_24120579' database. The table structure includes columns: PaymentID, PurchaseOrderID, PaymentDate, AmountPaid, PaymentMethod, and StaffID. The data shows six rows of payment records, each associated with a specific purchase order and staff member.

	PaymentID	PurchaseOrderID	PaymentDate	AmountPaid	PaymentMethod	StaffID
<input type="checkbox"/>	1	1	2025-01-12	222.40	Card	1000
<input type="checkbox"/>	2	2	2025-01-16	259.92	Bank Transfer	1001
<input type="checkbox"/>	3	3	2025-01-25	105.85	Card	1002
<input type="checkbox"/>	4	4	2025-02-12	734.23	Bank Transfer	1003
<input type="checkbox"/>	5	5	2025-03-10	31.05	Card	1004
<input type="checkbox"/>	6	6	2025-03-14	66.06	Bank Transfer	1005

Figure 9 - Payment Table

```
SELECT * FROM Payment;
```

Figure 9 shows the Payment table, which stores all the payments made for purchase orders. Details included are PaymentID, PurchaseOrderID, PaymentDate, AmountPaid, PaymentMethod and StaffID.

PurchaseOrderID and StaffID are foreign keys that link each payment to a related purchase order and the staff member who processed it.

The query below was used to calculate the total amount paid per purchase order:

```
SELECT PurchaseOrderLine.PurchaseOrderID,
       SUM(Product.UnitPrice * PurchaseOrderLine.Quantity) AS TotalCost
  FROM PurchaseOrderLine, Product
 WHERE PurchaseOrderLine.ProductID = Product.ProductID
 GROUP BY PurchaseOrderLine.PurchaseOrderID;
```

Task3

Introduction

This Task demonstrates the use of SQL queries to retrieve, update and display data for Parkside Purchase Order database. Each case focuses on different SQL statements, including INNER JOIN, WHERE, GROUP BY, UPDATE, DELETE and the COUNT function. These statements show how data can be searched, modified and grouped to help Parkside manage their day-to-day operations.

Case 1:

The query shows each purchase order in the database, showing the PurchaseOrderID, OrderDate and SupplierName. An INNER JOIN statement was used to join the data from the PurchaseOrder and Supplier tables, this allows for supplier details to be displayed alongside each other.

The screenshot shows the phpMyAdmin interface for a database named 'c8xpgso_24120579'. The current table is 'Payment'. The query executed is:

```
SELECT PurchaseOrder.PurchaseOrderID, PurchaseOrder.OrderDate, Supplier.Name AS SupplierName FROM PurchaseOrder INNER JOIN Supplier ON PurchaseOrder.SupplierID = Supplier.SupplierID;
```

The results show six rows of data:

PurchaseOrderID	OrderDate	SupplierName
1	2025-01-10	Medicare
2	2025-01-13	Discrem
3	2025-01-15	Vitality care
4	2025-02-10	Mediworld
5	2025-03-01	MK Medicals
6	2025-03-12	Excel Healthcare

Figure 10 - Case 1

```
SELECT PurchaseOrder.PurchaseOrderID, PurchaseOrder.OrderDate,  
Supplier.Name AS SupplierName  
  
FROM PurchaseOrder  
  
INNER JOIN Supplier  
  
ON PurchaseOrder.SupplierID = Supplier.SupplierID;
```

Figure 10 shows six purchase orders, showing its ID, order date and supplier connected to that order.

Case 2:

2a) The query lists all the products grouped by their supplier. The JOIN statement connects the Product and Supplier tables using the SupplierID. The GROUP BY statement is used to match each product to its suppliers, showing how many products each supplier provides.

The screenshot shows the phpMyAdmin interface with the following details:

- Left sidebar:** Shows the database structure with tables: c8xpgso_24120579, Delivery, Payment, Product, PurchaseOrder, PurchaseOrderLine, Staff, Supplier, information_schema, and performance_schema.
- Top navigation:** Server localhost:3306 > Database c8xpgso_24120579 > Table: Product
- Toolbar:** Browse, Structure, SQL, Search, Insert, Export, Import, Operations, Triggers.
- Query results:** A table showing the joined data from the Product and Supplier tables. The columns are SupplierName and ProductName. The data includes:

SupplierName	ProductName
Medicare	Disposable Syringes
Medicare	Examination Gloves
Medicare	Stethoscope
Discem	Blood Pressure Monitor
Discem	Bandages
Discem	Hand Sanitizer
Vitality care	Alcohol Swabs
Vitality care	Surgical Masks
Vitality care	Sharps Bin
Mediworld	Waiting Room Chairs
Mediworld	Glucose Testing Strips
Mediworld	Wall Clock
Mediworld	Otoscope Set
MK Medicals	Disinfectant Spray
MK Medicals	Tongue Depressors
MK Medicals	Floor Cleaner
Excel Healthcare	Digital Thermometer
Excel Healthcare	Office Printer Paper
Excel Healthcare	Clipboard

Figure 11 - Case 2a

```
SELECT Supplier.Name as SupplierName, Product.ProductName  
FROM Product  
JOIN Supplier ON Product.SupplierID = Supplier.SupplierID  
GROUP BY SupplierName, ProductName;
```

Figure 11 shows each supplier with the product they provided.

2b) This query searches for products based on their price. The WHERE statement is used to search for products that cost less than £50 as seen in Figure 12 . The same statement is used to find products that cost more than £50 in Figure 13.

The screenshot shows the phpMyAdmin interface with the 'Product' table selected. The query 'SELECT * FROM Product Where UnitPrice < 50;' has been run, returning 16 rows of data. The table includes columns: ProductID, ProductName, Description, UnitPrice, QuantityInStock, and SupplierID. The data shows various medical supplies and cleaning products with their respective details and prices.

	ProductID	ProductName	Description	UnitPrice	QuantityInStock	SupplierID
<input type="checkbox"/>	500	Disposable Syringes	Sterile 5ml syringes, box of 100	15.99	80	100
<input type="checkbox"/>	501	Examination Gloves	Nitrile gloves, medium, box of 100	12.50	12	100
<input type="checkbox"/>	503	Digital Thermometer	Fast-read digital thermometer	9.99	5	105
<input type="checkbox"/>	504	Alcohol Swabs	Isopropyl alcohol prep pads, 200 pcs	6.95	30	102
<input type="checkbox"/>	506	Bandages	Self-adhesive 4-inch roll bandages	4.99	58	101
<input type="checkbox"/>	507	Hand Sanitizer	500ml antibacterial gel bottles	3.50	60	101
<input type="checkbox"/>	508	Surgical Masks	3-ply disposable face masks, 50 pack	8.20	88	102
<input type="checkbox"/>	509	Office Printer Paper	A4 80gsm white, ream of 500 sheets	4.25	10	105
<input type="checkbox"/>	510	Disinfectant Spray	Surface cleaner, 750ml bottle	2.85	12	104
<input type="checkbox"/>	511	Waiting Room Chairs	Vinyl padded stackable chairs	45.00	24	103
<input type="checkbox"/>	512	Sharps Bin	5-litre yellow container for needles	7.95	45	102
<input type="checkbox"/>	513	Tongue Depressors	Wooden, non-sterile, box of 100	3.10	70	104
<input type="checkbox"/>	514	Glucose Testing Strips	50-count, for use with glucometer	14.50	100	103
<input type="checkbox"/>	515	Clipboard	A4 plastic with metal clip	2.20	12	105
<input type="checkbox"/>	516	Wall Clock	Silent, analog, battery-powered	11.75	2	103
<input type="checkbox"/>	518	Floor Cleaner	5-litre hospital-grade concentrate	16.80	34	104

Figure 12 - Case 2b (Products less than 50)

```
SELECT * FROM Product
```

```
Where UnitPrice < 50;
```

Figure 12 shows all products with a price less than £50

The screenshot shows the phpMyAdmin interface for the 'Product' table. The database is 'c8xpgso_24120579'. The table has columns: ProductID, ProductName, Description, UnitPrice, QuantityInStock, and SupplierID. The data is as follows:

ProductID	ProductName	Description	UnitPrice	QuantityInStock	SupplierID
502	Stethoscope	Professional grade dual-head	59.99	2	100
505	Blood Pressure Monitor	Upper arm automatic unit	75.00	12	101
517	Otoscope Set	Diagnostic otoscope with 3 specula	89.99	5	103

Figure 13 - Case 2b (Products greater than 50)

```
SELECT * FROM Product
```

```
Where UnitPrice > 50;
```

Figure 13 shows all products with a price greater than £50

Case 3

3a) This query updates the quantity of stock for Examination Gloves after a delivery. The UPDATE statement is used to increase the value of QuantityInStock by 50 WHERE the ProductID = 501, increasing the value from 12 to 62. Two SELECT statements were used before and after the update to show change in stock.

The screenshot shows the phpMyAdmin interface for the 'Product' table. The database is 'c8xpgso_24120579'. The table has columns: ProductID, ProductName, Description, UnitPrice, QuantityInStock, and SupplierID. The data is as follows:

ProductID	ProductName	Description	UnitPrice	QuantityInStock	SupplierID
501	Examination Gloves	Nitrile gloves, medium, box of 100	12.50	12	100

Figure 14 - Case 3a (Before Update)

```
SELECT * FROM Product WHERE ProductID = 501;
```

The screenshot shows the phpMyAdmin interface for a database named 'c8xpgso_24120579'. The 'Product' table is selected. The query history shows an UPDATE statement:

```
UPDATE Product SET QuantityInStock = QuantityInStock + 50 WHERE ProductID = 501;
```

The results of the SELECT query show one row affected:

```
SELECT * FROM Product WHERE ProductID = 501;
```

The table data shows the following row:

ProductID	ProductName	Description	UnitPrice	QuantityInStock	SupplierID
501	Examination Gloves	Nitrile gloves, medium, box of 100	12.50	62	100

Figure 15 – Case 3a (After Update)

```
UPDATE Product SET QuantityInStock = QuantityInStock + 50
```

```
WHERE ProductID = 501;
```

```
SELECT * FROM Product WHERE ProductID = 501;
```

Figure 14 shows the products original value of 12, while Figure 15 shows the updated value after 50 units were added.

3b) This query removes the Clipboard from the database. Before deleting the product, related records from the PurchaseOrderLine table were removed to avoid foreign key constraint errors. After that the product was removed from the product table using the DELETE statement. A SELECT statements were used to show that there was no data relating to ProductID = 515, and another SELECT statement was used to show that the product is no longer in the Product table.

ProductID	ProductName	Description	UnitPrice	QuantityInStock	SupplierID
515	Clipboard	A4 plastic with metal clip	2.20	12	105

Figure 16 - Case 3b (Before Delete)

```
SELECT * FROM Product WHERE ProductID = 515;
```

Figure 16 shows Clipboard before deletion.

The screenshot shows the phpMyAdmin interface for a database named 'c8xpgso_24120579'. The 'Product' table is selected. The SQL tab contains three queries:

- DELETE FROM PurchaseOrderLine WHERE ProductID = 515; (Result: 1 row deleted. (Query took 0.0079 seconds.))
- DELETE FROM Product WHERE ProductID = 515; (Result: 1 row deleted. (Query took 0.0075 seconds.))
- SELECT * FROM Product WHERE ProductID = 515; (Result: MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds.))

The table structure for 'Product' is shown below the queries:

ProductID	ProductName	Description	UnitPrice	QuantityInStock	SupplierID

At the bottom, there is a 'Query results operations' section with a 'Create view' button.

Figure 17 - Case 3b (After Delete)

```
DELETE FROM PurchaseOrderLine WHERE ProductID = 515;  
DELETE FROM Product WHERE ProductID = 515;  
SELECT * FROM Product WHERE ProductID = 515;
```

Figure 17 shows deletion of product and blank table when product is selected again.

The screenshot shows the phpMyAdmin interface for the 'Product' table in the 'c8xpgso_24120579' database. The table has columns: ProductID, ProductName, Description, UnitPrice, QuantityInStock, and SupplierID. The data includes various medical supplies and office equipment. The last row, ProductID 515, is highlighted in red, indicating it has been deleted.

	ProductID	ProductName	Description	UnitPrice	QuantityInStock	SupplierID
<input type="checkbox"/>	500	Disposable Syringes	Sterile 5ml syringes, box of 100	15.99	80	100
<input type="checkbox"/>	501	Examination Gloves	Nitrile gloves, medium, box of 100	12.50	62	100
<input type="checkbox"/>	502	Stethoscope	Professional grade dual-head	59.99	2	100
<input type="checkbox"/>	503	Digital Thermometer	Fast-read digital thermometer	9.99	5	105
<input type="checkbox"/>	504	Alcohol Swabs	Isopropyl alcohol prep pads, 200 pcs	6.95	30	102
<input type="checkbox"/>	505	Blood Pressure Monitor	Upper arm automatic unit	75.00	12	101
<input type="checkbox"/>	506	Bandages	Self-adhesive 4-inch roll bandages	4.99	58	101
<input type="checkbox"/>	507	Hand Sanitizer	500ml antibacterial gel bottles	3.50	60	101
<input type="checkbox"/>	508	Surgical Masks	3-ply disposable face masks, 50 pack	8.20	88	102
<input type="checkbox"/>	509	Office Printer Paper	A4 80gsm white, ream of 500 sheets	4.25	10	105
<input type="checkbox"/>	510	Disinfectant Spray	Surface cleaner, 750ml bottle	2.85	12	104
<input type="checkbox"/>	511	Waiting Room Chairs	Vinyl padded stackable chairs	45.00	24	103
<input type="checkbox"/>	512	Sharps Bin	5-litre yellow container for needles	7.95	45	102
<input type="checkbox"/>	513	Tongue Depressors	Wooden, non-sterile, box of 100	3.10	70	104
<input type="checkbox"/>	514	Glucose Testing Strips	50-count, for use with glucometer	14.50	100	103
<input type="checkbox"/>	516	Wall Clock	Silent, analog, battery-powered	11.75	2	103
<input type="checkbox"/>	517	Otoscope Set	Diagnostic otoscope with 3 specula	89.99	5	103
<input type="checkbox"/>	518	Floor Cleaner	5-litre hospital-grade concentrate	16.80	34	104

Figure 18 - Case 3b (Product Removed From Products)

```
SELECT * FROM Product;
```

Figure 18 shows complete Product table, without Clipboard (ProductID = 515), confirming product was successfully deleted.

Case 4

4a) This query updates the price of Disposable syringes to £20.00. The UPDATE statement changes the UnitPrice value in Product table where ProductID = 500. Two SELECT statements were used before and after to show change in price.

The screenshot shows the phpMyAdmin interface for the 'Product' table. The table has columns: ProductID, ProductName, and UnitPrice. A single row is selected for ProductID 500, which corresponds to 'Disposable Syringes' at a UnitPrice of 15.99. The 'Edit' button is highlighted, indicating the row is selected for modification.

Figure 19 - Case 4a (Before Update)

```
SELECT ProductID, ProductName, UnitPrice FROM Product WHERE ProductID = 500;
```

The screenshot shows the phpMyAdmin interface for the 'Product' table after the update. The UnitPrice for ProductID 500 has been changed to 20.00. The 'Edit' button is no longer highlighted, indicating the row is no longer selected.

Figure 20 - Case 4 (After Update)

```
UPDATE Product SET UnitPrice = 20.00 WHERE ProductID = 500;
```

```
SELECT ProductID, ProductName, UnitPrice FROM Product WHERE ProductID = 500;
```

Figure 19 show the original price of Disposable Syringes as £15.99, while Figure 20 shows the updated price of £20.00

4b)This query counts how many products were ordered in each purchase order. The COUNT() function is used to get the total number of products linked to each order in the PurchaseOrderLine table. The GROUP BY statement groups the results by PurchaseOrderID, showing one total for each order.

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** localhost:3306
- Database:** c8xpgso_24120579
- Table:** PurchaseOrderLine
- Query:**

```
SELECT PurchaseOrderID, COUNT(ProductID) AS NumberOfProducts FROM PurchaseOrderLine GROUP BY PurchaseOrderID;
```
- Results:**

PurchaseOrderID	NumberOfProducts
1	2
2	3
3	2
4	4
5	2
6	2
- Operations:** Print, Copy to clipboard, Export, Display chart, Create view.

Figure 21 - Case 4b

```
SELECT PurchaseOrderID, COUNT(ProductID) AS NumberOfProducts FROM
PurchaseOrderLine GROUP BY PurchaseOrderID;
```

Figure 21 shows each purchase order with the total number of products it contains.

4c) This query calculates the total amount spent on products from each supplier. JOIN statements are used to connect Supplier, product and PurchaseOrderLine tables. The SUM() function multiplies the unit price by the quantity for each product ordered to find the total cost and the GROUP BY statements groups the supplier by totals.

SupplierID	SupplierName	TotalAmountSpent
100	Medicare	262.50
101	Discern	259.92
102	Vitality care	105.85
103	Medworld	734.23
104	MK Medicals	31.05
105	Excel Healthcare	48.46

Figure 22 - Case 4c

```

SELECT Supplier.SupplierID, Supplier.Name AS SupplierName, SUM(Product.UnitPrice * PurchaseOrderLine.Quantity) AS TotalAmountSpent
FROM Supplier
JOIN Product ON Supplier.SupplierID = Product.SupplierID
JOIN PurchaseOrderLine ON Product.ProductID = PurchaseOrderLine.ProductID
GROUP BY Supplier.SupplierID, Supplier.Name;
    
```

Figure 22 shows the amount each supplier spent on their products.

References

- Bush, J. (2020). *Learn SQL Database Programming : Query and Manipulate Databases From Popular Relational Database Servers Using SQL* [online]. Birmingham: Packt Publishing.
- Connolly, T., & Begg, C. (2015). *Database Systems: A Practical Approach to Design, Implementation, and Management, Global Edition* [online]. Harlow: Pearson.
- Neagoie, A., & Binni, M. (n.d.). *Complete SQL and Databases Bootcamp*. Available from Udemy: <https://www.udemy.com/course/complete-sql-databases-bootcamp-zero-to-mastery/> [Accessed on 2025]
- Pajankar, A. (2020). *Learn SQL with MySQL* [online]. BPB Publications.
- Solomon, F., Jayaram, P., & Saqqa, A. A. (2019). *The SQL Workshop : Learn to Create, Manipulate and Secure Data and Manage Relational Databases with SQL* [online]. Packt Publishing.
- Ståle, H., and Refsnes, J. E., (1998). *SQL Tutorial*. Available from W3Schools: <https://www.w3schools.com/sql/> [Accessed on 2025]
- Venkataraman, R., Topi, H., and Hoffer, J. A., (2019). *Modern Database Management, Global Edition* [online]. Harlow: Pearson.

Appendix 1 – SQL Code

Task 2:

Create Tables

Staff:

```
CREATE TABLE Staff (
    StaffID INT AUTO_INCREMENT PRIMARY KEY,
    StaffFirstName VARCHAR(100),
    StaffLastName VARCHAR(100)
);
```

```
ALTER TABLE Staff AUTO_INCREMENT = 1000;
```

```
ALTER TABLE Staff CHANGE COLUMN StaffFirstName StafFirstName VARCHAR(100);
```

```
ALTER TABLE Staff CHANGE COLUMN StaffLastName StaffSurname VARCHAR(100);
```

```
ALTER TABLE Staff
MODIFY StafFirstName VARCHAR(100) NOT NULL,
MODIFY StaffSurname VARCHAR(100) NOT NULL;
```

Supplier:

```
CREATE TABLE Supplier (
    SupplierID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(100),
    ContactName VARCHAR(100),
    Phone VARCHAR(20),
    Email VARCHAR(100),
    Address VARCHAR(255)
);
```

```
ALTER TABLE Supplier AUTO_INCREMENT = 100;
```

```
ALTER TABLE Supplier
MODIFY Name VARCHAR(100) NOT NULL,
MODIFY ContactName VARCHAR(100) NOT NULL,
MODIFY Phone VARCHAR(20) NOT NULL,
MODIFY Email VARCHAR(100) UNIQUE NOT NULL,
MODIFY Address VARCHAR(255) NOT NULL;
```

Product:

```
CREATE TABLE Product (
    ProductID INT AUTO_INCREMENT PRIMARY KEY,
    ProductName VARCHAR(100),
    Description TEXT,
    UnitPrice DECIMAL(10,2) CHECK (UnitPrice > 0),
    QuantityInStock INT,
    SupplierID INT,
    CONSTRAINT fk_product_supplier FOREIGN KEY (SupplierID) REFERENCES
    Supplier(SupplierID)
);
```

```
ALTER TABLE Product AUTO_INCREMENT = 500;
```

```
ALTER TABLE Product
MODIFY ProductName VARCHAR(100) NOT NULL;
```

```
ALTER TABLE Product
MODIFY QuantityInStock INT CHECK (QuantityInStock >= 0);
```

PurchaseOrder:

```
CREATE TABLE PurchaseOrder (
    PurchaseOrderID INT AUTO_INCREMENT PRIMARY KEY,
    OrderDate DATE,
    StaffID INT,
    SupplierID INT,
    CONSTRAINT fk_po_staff FOREIGN KEY (StaffID) REFERENCES Staff(StaffID),
    CONSTRAINT fk_po_supplier FOREIGN KEY (SupplierID) REFERENCES
    Supplier(SupplierID)
);
```

```
ALTER TABLE PurchaseOrder
MODIFY OrderDate DATE NOT NULL;
```

```
ALTER TABLE PurchaseOrderLine
MODIFY Quantity INT CHECK (Quantity > 0);
```

PurchaseOrderLine:

```
CREATE TABLE PurchaseOrderLine (
    POLineID INT AUTO_INCREMENT PRIMARY KEY,
    PurchaseOrderID INT,
    ProductID INT,
    Quantity INT,
    CONSTRAINT fk_pol_order FOREIGN KEY (PurchaseOrderID) REFERENCES
    PurchaseOrder(PurchaseOrderID),
    CONSTRAINT fk_pol_product FOREIGN KEY (ProductID) REFERENCES
    Product(ProductID)
```

);

Delivery:

```
CREATE TABLE Delivery (
    DeliveryID INT AUTO_INCREMENT PRIMARY KEY,
    PurchaseOrderID INT,
    DeliveryDate DATE,
    StaffID INT,
    DeliveryStatus VARCHAR(50),
    CONSTRAINT fk_delivery_order FOREIGN KEY (PurchaseOrderID) REFERENCES
PurchaseOrder(PurchaseOrderID),
    CONSTRAINT fk_delivery_staff FOREIGN KEY (StaffID) REFERENCES Staff(StaffID)
);
```

```
ALTER TABLE Delivery
```

```
MODIFY DeliveryDate DATE NOT NULL,
MODIFY DeliveryStatus VARCHAR(50) NOT NULL;
```

Payment:

```
CREATE TABLE Payment (
    PaymentID INT AUTO_INCREMENT PRIMARY KEY,
    PurchaseOrderID INT,
    PaymentDate DATE,
    AmountPaid DECIMAL(10,2),
    PaymentMethod VARCHAR(50),
    StaffID INT,
    CONSTRAINT fk_payment_order FOREIGN KEY (PurchaseOrderID) REFERENCES
PurchaseOrder(PurchaseOrderID),
    CONSTRAINT fk_payment_staff FOREIGN KEY (StaffID) REFERENCES Staff(StaffID)
```

);

```
ALTER TABLE Payment  
MODIFY PaymentDate DATE NOT NULL,  
MODIFY AmountPaid DECIMAL(10,2) CHECK (AmountPaid >= 0),  
MODIFY PaymentMethod VARCHAR(50) NOT NULL;
```

Insert Data

Staff:

```
INSERT INTO Staff (StaffFirstName, StaffSurname) Values  
('Alice', 'Miller'),  
('John', 'Smith'),  
('Sarah', 'Brown'),  
('Michael', 'Davis'),  
('Emily', 'Johnson'),  
('Robert', 'Wilson');
```

Supplier:

```
INSERT INTO Supplier (Name, ContactName, Phone, Email, Address) VALUES  
(('Medicare','Oliver Queen','07760877635','oliver@medicare.com','44 Church Lane,  
Brighton'),  
(('Discem','Grant Gustin','07704335370','grant@discem.com','77 Richmond Road, South  
West London'),  
(('Vitality care','Walter Black','07019811309','walter@vitalitycare.com','12 Victoria Road, East  
London'),  
(('Mediworld','Gavin Booth','07856473374','gavin@mediworld','838 Kingsway, Hemel  
Hempstead'),
```

('MK Medicals','Russell Roman','07859634762','russell@mkmedical.com','59 George Street, Darlington'),

('Excel Healthcare','David Kirby','07702214012','david@excelhealthcare','99 Station Road, Glasgow');

Product:

```
INSERT INTO Product (ProductName, Description, UnitPrice, QuantityInStock) VALUES  
('Disposable Syringes','Sterile 5ml syringes, box of 100', 15.99 , 80),  
('Examination Gloves ','Nitrile gloves, medium, box of 100', 12.50, 12),  
('Stethoscope','Professional grade dual-head', 59.99, 2),  
('Digital Thermometer','Fast-read digital thermometer', 9.99 , 5),  
('Alcohol Swabs','Isopropyl alcohol prep pads, 200 pcs', 6.95, 30),  
('Blood Pressure Monitor','Upper arm automatic unit', 75.00, 12),  
('Bandages','Self-adhesive 4-inch roll bandages', 4.99, 58),  
('Hand Sanitizer','500ml antibacterial gel bottles', 3.50, 60),  
('Surgical Masks','3-ply disposable face masks, 50 pack', 8.20, 88),  
('Office Printer Paper','A4 80gsm white, ream of 500 sheets', 4.25, 10),  
('Disinfectant Spray','Surface cleaner, 750ml bottle', 2.85, 12),  
('Waiting Room Chairs','Vinyl padded stackable chairs', 45.00, 24),  
('Sharps Bin','5-litre yellow container for needles', 7.95, 45),  
('Tongue Depressors','Wooden, non-sterile, box of 100', 3.10, 70),  
('Glucose Testing Strips','50-count, for use with glucometer', 14.50, 100),  
('Clipboard','A4 plastic with metal clip', 2.20, 12),  
('Wall Clock','Silent, analog, batterypowered', 11.75, 2),  
('Otoscope Set','Diagnostic otoscope with 3 specula', 89.99, 5),  
('Floor Cleaner','5-litre hospital-grade concentrate', 16.80, 34);
```

```
UPDATE Product SET SupplierID = 100 WHERE ProductID IN (500, 501);
UPDATE Product SET SupplierID = 101 WHERE ProductID IN (505, 507, 506);
UPDATE Product SET SupplierID = 102 WHERE ProductID IN (508, 512);
UPDATE Product SET SupplierID = 103 WHERE ProductID IN (511, 516, 517, 514);
UPDATE Product SET SupplierID = 104 WHERE ProductID IN (510, 518);
UPDATE Product SET SupplierID = 105 WHERE ProductID IN (503, 509, 515);
UPDATE Product SET SupplierID = 100 WHERE ProductID IN (502);
UPDATE Product SET SupplierID = 102 WHERE ProductID IN (504);
UPDATE Product SET SupplierID = 104 WHERE ProductID IN (513);
```

Purchase Order:

```
INSERT INTO PurchaseOrder (OrderDate, StaffID, SupplierID) VALUES
('2025-01-10', 1000, 100),
('2025-01-13', 1001, 101),
('2025-01-15', 1002, 102),
('2025-02-10', 1003, 103),
('2025-03-01', 1004, 104),
('2025-03-12', 1005, 105);
```

Purchase Order Line:

```
INSERT INTO PurchaseOrderLine (PurchaseOrderID, ProductID, Quantity) VALUES  
    -- Order 1  
    (1, 500, 10),  
    (1, 501, 5),  
    -- Order 2  
    (2, 505, 2),  
    (2, 507, 20),  
    (2, 506, 8),  
    -- Order 3  
    (3, 508, 10),  
    (3, 512, 3),  
    -- Order 4  
    (4, 511, 4),  
    (4, 516, 1),  
    (4, 517, 2),  
    (4, 514, 25),  
    -- Order 5  
    (5, 510, 5),  
    (5, 518, 1),  
    -- Order 6  
    (6, 503, 4),  
    (6, 509, 2),  
    (6, 515, 8);
```

Delivery:

```
INSERT INTO Delivery (PurchaseOrderID, DeliveryDate, StaffID, DeliveryStatus) VALUES  
(1, '2025-01-12', 1000, 'Complete'),  
(2, '2025-01-16', 1001, 'Complete'),  
(3, '2025-01-25', 1002, 'Complete'),  
(4, '2025-02-12', 1003, 'Partial'),  
(5, '2025-03-10', 1004, 'Complete'),  
(6, '2025-03-14', 1005, 'Returned');
```

Payment:

```
SELECT PurchaseOrderLine.PurchaseOrderID,  
SUM(Product.UnitPrice * PurchaseOrderLine.Quantity) AS TotalCost  
FROM PurchaseOrderLine, Product  
WHERE PurchaseOrderLine.ProductID = Product.ProductID  
GROUP BY PurchaseOrderLine.PurchaseOrderID;
```

```
INSERT INTO Payment (PurchaseOrderID, PaymentDate, AmountPaid, PaymentMethod,  
StaffID) VALUES  
(1, '2025-01-12', 222.40, 'Card', 1000),  
(2, '2025-01-16', 259.92, 'Bank Transfer', 1001),  
(3, '2025-01-25', 105.85, 'Card', 1002),  
(4, '2025-02-12', 734.23, 'Bank Transfer', 1003),  
(5, '2025-03-10', 31.05, 'Card', 1004),  
(6, '2025-03-14', 66.06, 'Bank Transfer', 1005);
```