

## Scenario:

I am creating a database for a chemist / pharmacy. The database contains product, customer and doctors surgery information, with queries, in-built functions and stored procedures that enable pharmacy staff to easily access information.

The product information will populate 2 tables, one with basic product info called 'Products' and a second table with medicinal product info called 'Product\_Specs'.

The 'Products' table has a default value for the 'Price' column. This is set to the cost of a UK prescription, so that if no price is entered this will autopopulate.

The 'Product\_Specs' table has easily accessible safety / medical information in case a customer has a query or a product is prescribed to someone of the wrong age, etc.

The database will also have a stock table. This will show the quantity of each unit of product in the pharmacy. The stock table will also contain delivery time in days for each product when the pharmacy requests it from the manufacturers.

The customer info is also split into 2 tables, one with relevant customer information called 'Customers' and another with the customer contact details called 'Customers\_Contact'.

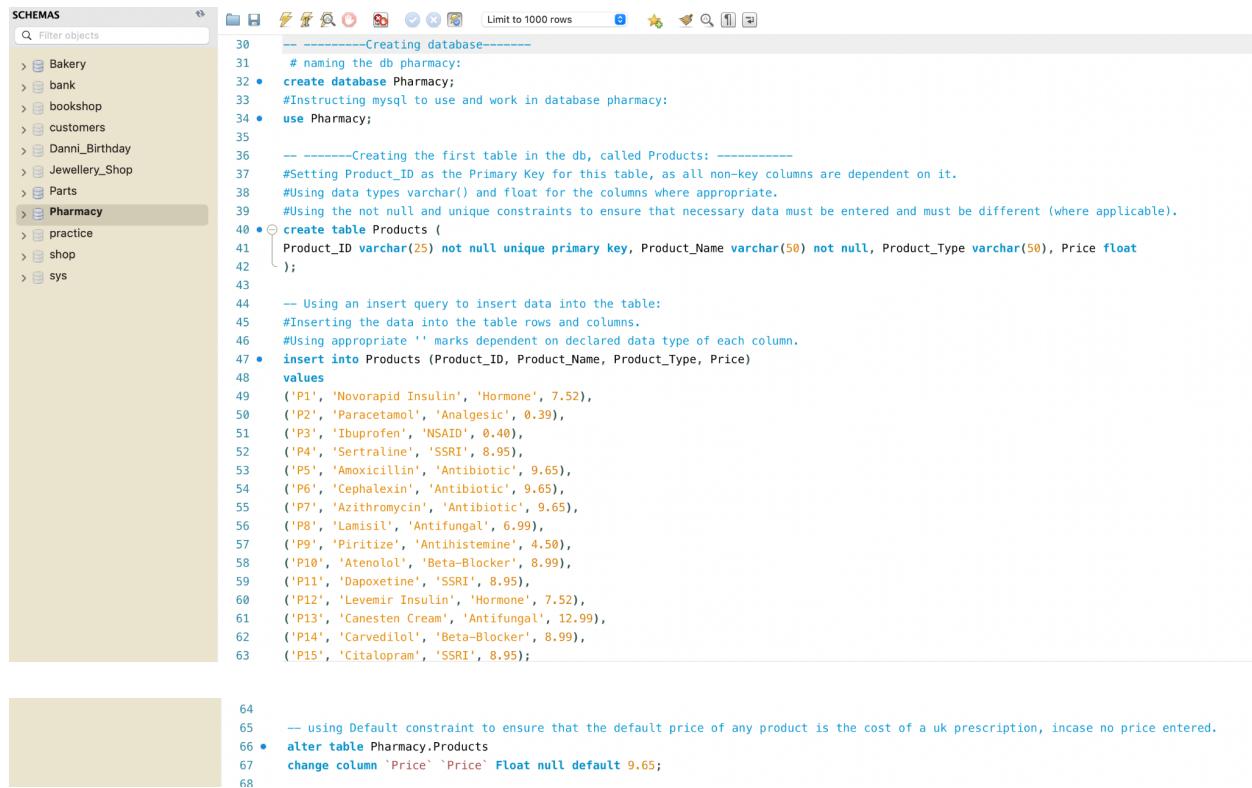
The 'Customer' table has the relevant information that the pharmacy needs for issuing prescriptions, such as customer age and any existing medical conditions that may effect the medication they are prescribed or buying.

The Doctor's Surgery tables are split into 2 tables, one providing a Surgery\_ID to each practice to maintain db normalisation, and the second table contains the contact information for each surgery and is called 'Surgery\_Info'.

There is a final table called 'Cust\_Surgery', which contains a list of which customers are registered to which surgery. This way if there is ever an issue with a prescription for a patient, the pharmacy can easily query which Drs surgery to contact.

# Creating Pharmacy Database and tables:

## Creating first table:



The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' tree view is open, showing various databases like Bakery, bank, bookshop, customers, Danni\_Birthday, Jewellery\_Shop, Parts, and Pharmacy. The Pharmacy database is selected. The main pane displays a SQL script for creating a database and a table, with some lines highlighted in yellow.

```
30  -----Creating database-----
31  > # naming the db pharmacy;
32 •  create database Pharmacy;
33  #Instructing mysql to use and work in database pharmacy:
34 •  use Pharmacy;
35
36  -----Creating the first table in the db, called Products: -----
37  #Setting Product_ID as the Primary Key for this table, as all non-key columns are dependent on it.
38  #Using data types varchar() and float for the columns where appropriate.
39  #Using the not null and unique constraints to ensure that necessary data must be entered and must be different (where applicable).
40 •  create table Products (
41      Product_ID varchar(25) not null unique primary key, Product_Name varchar(50) not null, Product_Type varchar(50), Price float
42  );
43
44  -- Using an insert query to insert data into the table:
45  #Inserting the data into the table rows and columns.
46  #Using appropriate '' marks dependent on declared data type of each column.
47 •  insert into Products (Product_ID, Product_Name, Product_Type, Price)
48  values
49  ('P1', 'Novorapid Insulin', 'Hormone', 7.52),
50  ('P2', 'Paracetamol', 'Analgesic', 0.39),
51  ('P3', 'Ibuprofen', 'NSAID', 0.40),
52  ('P4', 'Sertraline', 'SSRI', 8.95),
53  ('P5', 'Amoxicillin', 'Antibiotic', 9.65),
54  ('P6', 'Cephalexin', 'Antibiotic', 9.65),
55  ('P7', 'Azithromycin', 'Antibiotic', 9.65),
56  ('P8', 'Lamisil', 'Antifungal', 6.99),
57  ('P9', 'Piritizite', 'Antihistamine', 4.50),
58  ('P10', 'Atenolol', 'Beta-Blocker', 8.99),
59  ('P11', 'Dapoxetine', 'SSRI', 8.95),
60  ('P12', 'Levemir Insulin', 'Hormone', 7.52),
61  ('P13', 'Canesten Cream', 'Antifungal', 12.99),
62  ('P14', 'Carvedilol', 'Beta-Blocker', 8.99),
63  ('P15', 'Citalopram', 'SSRI', 8.95);
64
65  -- using Default constraint to ensure that the default price of any product is the cost of a uk prescription, incase no price entered.
66 •  alter table Pharmacy.Products
67  change column `Price` `Price` Float null default 9.65;
```

## Output:

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' tree view has the 'Pharmacy' schema selected. In the main pane, a SQL editor window displays the following code and its results:

```
62 ('P14', 'Carvedilol', 'Beta-Blocker', 8.99),
63 ('P15', 'Citalopram', 'SSRI', 8.95);

select * from Products;
```

The result grid shows the following data:

Product_ID	Product_Name	Product_Type	Price
P1	Novorapid Insulin	Hormone	7.52
P10	Atenolol	Beta-Blocker	8.99
P11	Dapoxetine	SSRI	8.95
P12	Levemir Insulin	Hormone	7.52
P13	Canesten Cream	Antifungal	12.99
P14	Carvedilol	Beta-Blocker	8.99
P15	Citalopram	SSRI	8.95
P2	Paracetamol	Analgesic	0.39
P3	Ibuprofen	NSAID	0.4
P4	Sertraline	SSRI	8.95
P5	Amoxicillin	Antibiotic	9.65
P6	Cephalexin	Antibiotic	9.65
P7	Azithromycin	Antibiotic	9.65
P8	Lamisil	Antifungal	6.99
P9	Piritazine	Antihistamine	9.12
<b>NULL</b>	<b>NULL</b>	<b>NULL</b>	<b>NULL</b>

## Creating second table, 'Product\_Specs':

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' tree view has the 'Pharmacy' schema selected. In the main pane, a SQL editor window displays the following code:

```
68 -- -----Creating the second table in the db, called Products_specs: -----
69 #This table has all important medicinal information pertaining to the products in table product.
70 #Setting Product_ID as the Foreign Key to connect the tables.
71 #Using the data types varchar() and int.
72 #Using constraints unique and not null where appropriate.
73
74 • create table Product_Specs (
75   Product_ID varchar(25) not null unique, Active_Ingredient varchar(50), Min_Age_Yrs int, Manufacturer varchar(50),
76   foreign key(Product_ID) references Products(Product_ID)
77 );
78 -- Using an insert query to insert data into the table:
79 • insert into Product_Specs(
80   Product_ID, Active_Ingredient, Min_Age_Yrs, Manufacturer)
81   values
82   ('P1', 'Human Insulin', 0, 'Novo-Nordisk'),
83   ('P2', 'Acetaminophen', 6, 'Aspar Pharma'),
84   ('P3', 'Ibuprofen', 8, 'Aspar Pharma'),
85   ('P4', 'Sertraline', 6, 'Amarox Ltd'),
86   ('P5', 'Trihydrate', 0, 'GlaxoSmithKline'),
87   ('P6', 'Cephalexin monohydrate', 0, 'GlaxoSmithKline'),
88   ('P7', 'Azithromycin dihydrate', 0, 'GlaxoSmithKline'),
89   ('P8', 'Terbinafine hydrochloride', 1, 'Karo Healthcare'),
90   ('P9', 'Cetirizine Hydrochloride', 6, 'Amarox Ltd'),
91   ('P10', 'Atenolol', 18, 'GlaxoSmithKline'),
92   ('P11', 'Dapoxetine hydrochloride salt', 18, 'Amarox Ltd'),
93   ('P12', 'Insulin detemir', 0, 'Novo-Nordisk'),
94   ('P13', 'Clotrimazole', 12, 'Bayer Pharmaceuticals'),
95   ('P14', 'Carvedilol AN', 2, 'GlaxoSmithKline'),
96   ('P15', 'Citalopram hydrobromide', 18, 'Amarox Ltd');
97
98 • select * from Product_Specs;
99
100
```

## Output:

SCHEMAS

Filter objects
Result Grid
Filter Rows:
Search
Edit:
Export/Import:

	Product_ID	Active_Ingredient	Min_Age_Yrs	Manufacturer
>	P1	Human Insulin	0	Novo-Nordisk
>	P10	Atenolol	18	GlaxoSmithKline
>	P11	Dapoxetine hydrochloride salt	18	Amarox Ltd
>	P12	Insulin detemir	0	Novo-Nordisk
>	P13	Clotrimazole	12	Bayer Pharma...
>	P14	Carvedilol AN	2	GlaxoSmithKline
>	P15	Citalopram hydrobromide	18	Amarox Ltd
>	P2	Acetaminophen	6	Aspar Pharma
>	P3	Ibuprofen	8	Aspar Pharma
>	P4	Sertraline	6	Amarox Ltd
>	P5	Trihydrate	0	GlaxoSmithKline
>	P6	Cephalexin monohydrate	0	GlaxoSmithKline
>	P7	Azithromycin dihydrate	0	GlaxoSmithKline
>	P8	Berbainafine hydrochloride	1	Karo Healthcare
>	P9	Cetirizine Hydrochloride	6	Amarox Ltd
	NULL	NULL	NULL	NULL

## Creating third table, ‘Stock’:

SCHEMAS

Filter objects

Bakery  
bank  
bookshop  
customers  
Danni\_Birthday  
Jewellery\_Shop  
Parts  
**Pharmacy**  
practice  
shop  
sys

103 -- -----Creating the third table in the db, called Stock: -----  
104 #This table contains data about the quantity of stock of each product in the pharmacy.  
105 #The Product\_ID is set as the PK as all other non-key attributes depend on it.  
106 #The data types used here are varchar() and int.  
107 #Using not null and unique constraints where appropriate.  
108  
109 • create table Stock (  
110     Product\_ID varchar(25) not null unique, No\_of\_Units int not null, DeliveryTime\_Days int);  
111  
112 -- Using an insert query to insert data into the table:  
113 • insert into Stock(  
114     Product\_ID, No\_of\_Units, DeliveryTime\_Days)  
115     values  
116     ('P1', 30, 5),  
117     ('P2', 35, 3),  
118     ('P4', 31, 2),  
119     ('P5', 24, 4),  
120     ('P6', 32, 3),  
121     ('P7', 25, 5),  
122     ('P8', 39, 4),  
123     ('P9', 32, 2),  
124     ('P10', 41, 2),  
125     ('P11', 12, 3),  
126     ('P12', 15, 5),  
127     ('P13', 11, 6),  
128     ('P14', 17, 4),  
129     ('P15', 28, 3);  
130  
131 #Using alter to place Product ID as the foreign key here again to link the tables together:  
132 • alter table Stock  
133     add constraint Product\_ID\_FK  
134     foreign key(Product\_ID) references Products(Product\_ID);  
135

## Output:

Result Grid			
	Product_ID	No_of_Units	DeliveryTime_Days
>	P1	30	5
	P10	41	2
	P11	12	3
	P12	15	5
	P13	11	6
	P14	17	4
	P15	28	3
	P2	35	3
	P4	31	2
	P5	24	4
	P6	32	3
	P7	25	5
	P8	39	4
	P9	32	2
	HULL	HULL	HULL

## Creating fourth table, ‘Customers’:

SCHEMAS

The screenshot shows the MySQL Workbench interface with the 'SCHEMAS' tree on the left. The 'Pharmacy' schema is selected. In the main pane, a script is displayed for creating the 'Customers' table and inserting data. The code is as follows:

```
136 -- -----Creating the fourth table in the db, called Customers: -----
137 #This table contains information about the customers of the pharmacy.
#The Customer_ID is set as the PK as each non-key attribute is dependent on this column.
#Using varchar() and a fourth data type, date, where appropriate for the content of the columns of this table.
138 create table Customers(
    Customer_ID varchar(25) not null unique primary key, First_Name varchar(50) not null, Surname varchar(50) not null, DOB date, Known_Conditions
);

139 -- Using an insert query to insert data into the table:
140 insert into Customers(
    Customer_ID, First_Name, Surname, DOB, Known_Conditions
)
141     values
142 ('C1', 'Rachel', 'Greene', '1968-08-02', 'T1 Diabetes'),
('C2', 'Monica', 'Bing', '1968-12-03', 'OCD'),
('C3', 'Chandler', 'Bing', '1967-04-09', 'Eczema'),
('C4', 'Joey', 'Tribbiani', '1965-01-07', 'T2 Diabetes'),
('C5', 'Ross', 'Geller', '1962-06-09', 'Asthma'),
('C6', 'Pheobe', 'Buffay', '1969-07-08', 'Anxiety'),
('C7', 'Gunther', 'Smith', '1963-10-04', 'hypertension'),
('C8', 'Deanna', 'Trol', '1958-11-25', 'Epilepsy'),
('C9', 'William', 'Ricker', '1955-01-31', 'Athritis'),
('C10', 'Jean-Luc', 'Picard', '1940-05-02', 'Anaemia'),
('C11', 'Beverly', 'Crusher', '1952-09-07', 'Depression'),
('C12', 'Geordi', 'Laforge', '1951-03-10', 'Glaucoma');
```

## Output:

SCHEMAS

Result Grid

Customer_ID	First_Name	Surname	DOB	Known_Conditions
C1	Rachel	Greene	1968-08-02	T1 Diabetes
C10	Jean-Luc	Picard	1940-05-02	Anemia
C11	Beverly	Crusher	1952-09-07	Depression
C12	Geordi	Laforge	1951-03-10	Glaucoma
C2	Monoica	Bing	1968-12-03	OCD
C3	Chandler	Bing	1967-04-09	Eczema
C4	Joey	Tribbiani	1965-01-07	T2 Diabetes
C5	Ross	Geller	1962-06-09	Ashtma
C6	Pheobe	Buffay	1969-07-08	Aniety
C7	Gunther	Smith	1963-10-04	hypertension
C8	Deanna	Troi	1958-11-25	Epilepsy
C9	William	Ricker	1955-01-31	Athritis
NULL	NULL	NULL	NULL	NULL

## Creating fifth table, 'Customers\_Contact':

```

SCHEMAS
Result Grid
Limit to 1000 rows
159 ('C12', 'Geordi', 'Laforge', '1951-03-10', 'Glaucoma');
160
161 -- -----Creating the fifth table in the db, called Customers_Contact: -----
162 #Creating this table with the contact information for each customer.
163 #Using varchar() for all the columns.
164 #For customer phone numbers i have used varchar instead of int in case any number needs a country code (i.e.+33 or any hyphens).
165 #I am using the unique and not null constraints.
166 • create table Customers_Contact (
167   Customer_ID varchar(25) not null unique, Phone varchar(11), Email varchar(50), Address varchar(50));
168
169 -- Using an insert query to insert data into the table:
170 • insert into Customers_Contact(
171   Customer_ID, Phone, Email, Address)
172   values
173   ('C1', '02035079861', 'RG@gmail.com', '512 Central Rd, London, KT7 4DU'),
174   ('C2', '07865485973', 'Monica.Bing@outlook.com', '495 Grove St, London, KT3 6DU'),
175   ('C3', '07685943491', 'Chandler.B@hotmail.com', '495 Grove St, London, KT3 6DU'),
176   ('C4', '07848210365', 'JTacting@gmail.com', '112 Peru Ave, London, KT9, 3BU'),
177   ('C5', '07986525326', 'dr.geller@outlook.com', '208 City St, London, KT8 1DE'),
178   ('C6', '07875262895', 'PheobsBuffay1@aol.com', '12 Bush Rd, London, KT9 1DU'),
179   ('C7', '07656392914', 'Gunther@centralperkcafe.com', '52 Halt Way, London, KT3 2UD'),
180   ('C8', '07858654568', 'Counselor.Troi@Federation.com', '312 Star Rd, London, KT12, 5DU'),
181   ('C9', '07646285393', 'will.ricker1@enterprise.com', '82 Galaxy Ave, London KT1, 9FP'),
182   ('C10', '07652585965', 'capticpicard@starship.com', '91 Fleet St, London, KT9 0PL'),
183   ('C11', '07959632381', 'Dr.Crusher@gmail.com', '11 Rose Rd, London, KT7 1PD'),
184   ('C12', '07545859626', 'Geordi.Laforge@thehelm.com', '58 Brick Lane, London, KT1 9UL');
185
186 #Using alter to make Customer_ID the Foreign Key to link the tables together.
187 • alter table Customers_Contact
188   add constraint Customer_ID_FK
189   foreign key (Customer_ID) references customers(Customer_ID);
...

```

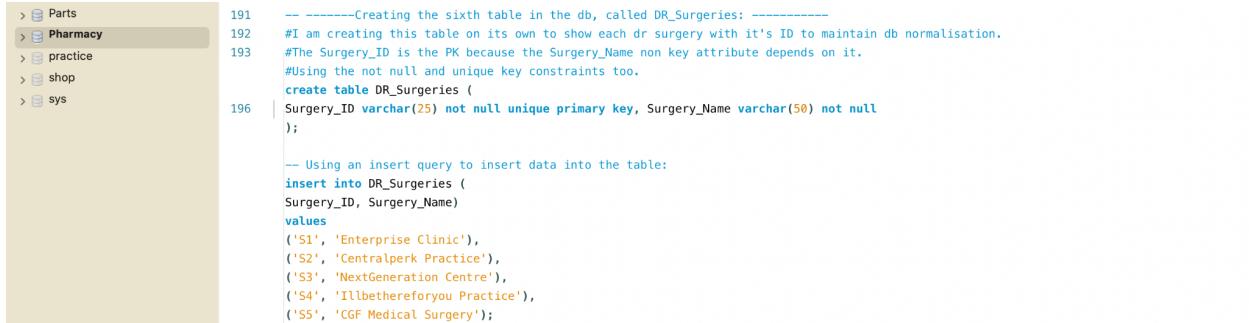
## Output:

SCHEMAS

Result Grid

Customer_ID	Phone	Email	Address
C1	02035079861	RG@gmail.com	512 Central Rd, London, KT7 4DU
C10	07652585965	capticpicard@starship.com	91 Fleet St, London, KT9 0PL
C11	07959632381	Dr.Crusher@gmail.com	11 Rose Rd, London, KT7 1PD
C12	07545859626	Geordi.Laforge@thehelm.com	58 Brick Lane, London, KT1 9UL
C2	07865485973	Monica.Bing@outlook.com	495 Grove St, London, KT3 6DU
C3	07685943491	Chandler.B@hotmail.com	495 Grove St, London, KT3 6DU
C4	07848210365	JTacting@gmail.com	112 Peru Ave, London, KT9, 3BU
C5	07986525326	dr.geller@outlook.com	208 City St, London, KT8 1DE
C6	07875262895	PheobsBuffay1@aol.com	12 Bush Rd, London, KT9 1DU
C7	07656392914	Gunther@centralperkcafe.com	52 Halt Way, London, KT3 2UD
C8	07858654568	Counselor.Troi@Federation...	312 Star Rd, London, KT12, 5DU
C9	07646285393	will.ricker1@enterprise.com	82 Galaxy Ave, London KT1, 9FP
NULL	NULL	NULL	NULL

## Creating sixth table, ‘DR\_Surgeries’:

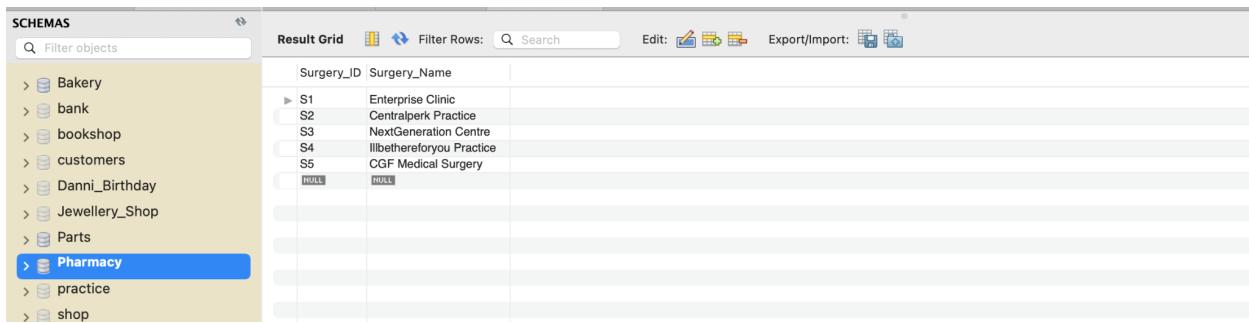


```
> Parts
> Pharmacy
> practice
> shop
> sys

191 -- -----Creating the sixth table in the db, called DR_Surgeries: -----
192 #I am creating this table on its own to show each dr surgery with it's ID to maintain db normalisation.
193 #The Surgery_ID is the PK because the Surgery_Name non key attribute depends on it.
194 #Using the not null and unique key constraints too.
195 create table DR_Surgeries (
196     Surgery_ID varchar(25) not null unique primary key, Surgery_Name varchar(50) not null
197 );

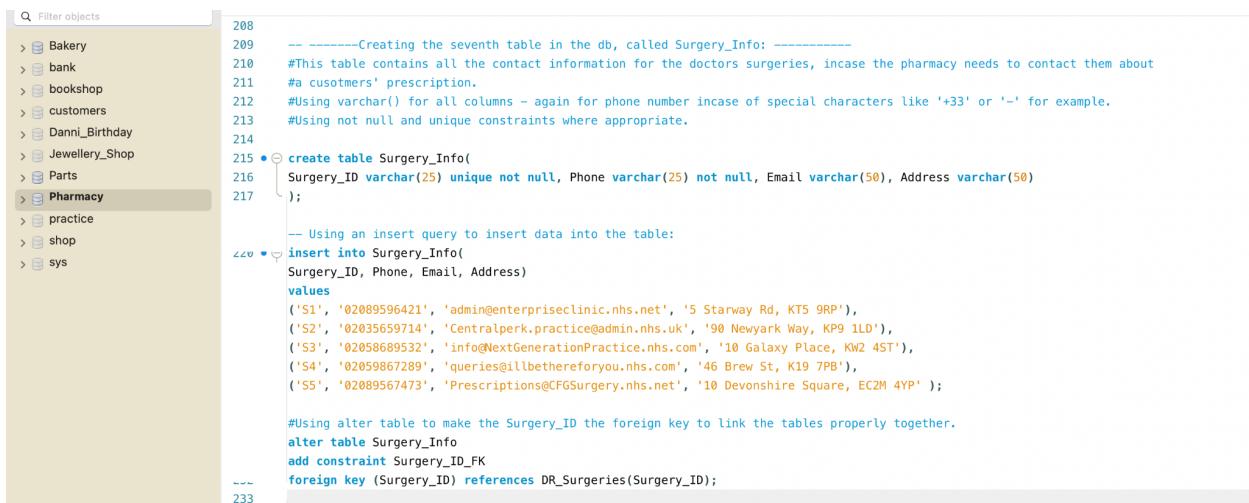
-- Using an insert query to insert data into the table:
198 insert into DR_Surgeries (
199     Surgery_ID, Surgery_Name
200     values
201     ('S1', 'Enterprise Clinic'),
202     ('S2', 'Centralperk Practice'),
203     ('S3', 'NextGeneration Centre'),
204     ('S4', 'Illbethereforyou Practice'),
205     ('S5', 'CGF Medical Surgery');
```

## Output:



Surgery_ID	Surgery_Name
S1	Enterprise Clinic
S2	Centralperk Practice
S3	NextGeneration Centre
S4	Illbethereforyou Practice
S5	CGF Medical Surgery
NULL	NULL

## Creating seventh table, ‘Surgery\_Info’:



```
> Parts
> Pharmacy
> practice
> shop
> sys

208 -- -----Creating the seventh table in the db, called Surgery_Info: -----
209 #This table contains all the contact information for the doctors surgeries, incase the pharmacy needs to contact them about
210 #a cusotmers' prescription.
211 #Using varchar() for all columns – again for phone number incase of special characters like '+33' or '-' for example.
212 #Using not null and unique constraints where appropriate.
213
214
215 • create table Surgery_Info(
216     Surgery_ID varchar(25) unique not null, Phone varchar(25) not null, Email varchar(50), Address varchar(50)
217 );
218
219
220 -- Using an insert query to insert data into the table:
221 insert into Surgery_Info(
222     Surgery_ID, Phone, Email, Address
223     values
224     ('S1', '02089596421', 'admin@enterpriseclinic.nhs.net', '5 Starway Rd, KT5 9RP'),
225     ('S2', '02035659714', 'Centralperk.practice@admin.nhs.uk', '90 Newyark Way, KP9 1LD'),
226     ('S3', '02058689532', 'info@NextGenerationPractice.nhs.com', '10 Galaxy Place, KW2 4ST'),
227     ('S4', '02059867289', 'queries@illbethereforyou.nhs.com', '46 Brew St, K19 7PB'),
228     ('S5', '02089567473', 'Prescriptions@CGFSurgery.nhs.net', '10 Devonshire Square, EC2M 4YP' );
229
230
231 #Using alter table to make the Surgery_ID the foreign key to link the tables properly together.
232 alter table Surgery_Info
233 add constraint Surgery_ID_FK
234 foreign key (Surgery_ID) references DR_Surgeries(Surgery_ID);
```

## Output:

SCHEMAS

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

Surgery_ID	Phone	Email	Address
S1	02089596421	admin@enterpriseclinic.nhs.net	5 Starway Rd, KT5 9RP
S2	02035659714	Centralperk.practice@admin.nhs.uk	90 Newyark Way, KP9 1LD
S3	02058689532	info@NextGenerationPractice.nhs.com	10 Galaxy Place, KW2 4ST
S4	02059867289	queries@ilbetherethereyou.nhs.com	46 Brew St, K19 7PB
S5	02089567473	Prescriptions@CFGSurgery.nhs.net	10 Devonshire Square, EC2M 4YP
HULL	HULL	HULL	HULL

## Creating eighth table, 'Cust\_Surgery':

```

234 -- -----Creating the eighth table in the db, called Cust_Surgery: -----
235 #This is a table that contains shows which customer is registered to which doctors surgery.
236 #This way the pharmacy knows which doctors surgery to contact if there is a prescription issue or query.
237 #The patient ID is the PK.
238 #Using the varchar() data type.
239 #Using the not null and unique constraints.
240 create table Cust_Surgery(
241 Patient_ID varchar(25) unique not null primary key, Customer_ID Varchar(25) not null, Surgery_ID varchar (25) not null
242 );
243
244 -- Using an insert query to insert data into the table:
245 #Inserting values into this table:
246 insert into Cust_Surgery(
247 Patient_ID, Customer_ID, Surgery_ID)
248 values
249 ('Pt1', 'C1', 'S3'),
('Pt2', 'C2', 'S5'),
('Pt3', 'C3', 'S1'),
('Pt4', 'C4', 'S2'),
('Pt5', 'C5', 'S4'),
('Pt6', 'C6', 'S3'),
('Pt7', 'C7', 'S4'),
('Pt8', 'C8', 'S1'),
('Pt9', 'C9', 'S5'),
('Pt10', 'C10', 'S5'),
('Pt11', 'C11', 'S2'),
('Pt12', 'C12', 'S2');
250
251 #Using the alter table to make both Customer_ID and Surgery_ID foreign keys in this table to link them to their respective other
252 #tables (which are DR_Surgery and Customers).
253 alter table Cust_Surgery
254 add constraint Customer_ID_FK2
255 foreign key (Customer_ID) references Customers(Customer_ID);
256
257
258
259
260
261
262
263
264
265
266
267
268
269

```

## Output:

SCHEMAS

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

Patient_ID	Customer_ID	Surgery_ID
Pt1	C1	S3
Pt10	C10	S5
Pt11	C11	S2
Pt12	C12	S2
Pt2	C2	S5
Pt3	C3	S1
Pt4	C4	S2
Pt5	C5	S4
Pt6	C6	S3
Pt7	C7	S4
Pt8	C8	S1
Pt9	C9	S5
HULL	HULL	HULL

## Using queries to retrieve data:

### Query1 code:

```
-----Using 5 queries to retrieve data: -----
#Creating a query to return all products sold at the pharmacy that are manufactured by GlaxoSmithKline, ordered by Product ID.
#This would be a useful query in case of a manufacturer recall of products, for example.
select ps.Product_ID
from Product_Specs ps
where Manufacturer = 'GlaxoSmithKline'
order by Product_ID; #Ordering by product_id.
#Please note: because I have used varchar() for Product_ID column it runs the values as strings,
#and so 'P10', 'P11' etc, will be ordered above 'P2'.
```

### Output:

Product_ID
P10
P14
P5
P6
P7
NULL

### Query 2 code and output:

Product_Type
SSRI
NSAID
Hormone
Beta-Blocker
Antihistamine
Antifungal
Antibiotic
Analgesic

## Query 3 code and output:

The screenshot shows a database interface with a sidebar containing a tree view of schema objects. The 'Pharmacy' schema is selected, and under it, the 'Tables' node is expanded, showing 'Customers', 'Customers\_Contact', 'Cust\_Surgery', 'DR\_Surgeries', 'Products', 'Product\_Specs', 'Stock', and 'Surgery\_Info'. The main pane displays the following SQL code:

```
290
291 #Creating a query to retrieve the name of the products and then sort the products by price.
292 #This would be a useful query for accounting departments of the pharmacy.
293 • select pr.Product_Name, pr.Price
294   from Products pr
295   Order by Price;
296
297
```

Below the code is a 'Result Grid' table with columns 'Product\_Name' and 'Price'. The data is as follows:

Product_Name	Price
Paracetamol	0.39
Ibuprofen	0.4
Lamisil	6.99
Novorapid Insulin	7.52
Levemir Insulin	7.52
Dapoxetine	8.95
Citalopram	8.95
Sertraline	8.95
Atenolol	8.99
Carvedilol	8.99
Pritize	9.12
Amoxicillin	9.65
Cephalexin	9.65
Azithromycin	9.65
Canesten Cream	12.99

## Query 4 code and output:

The screenshot shows a database interface with a sidebar containing a tree view of schema objects. The 'Pharmacy' schema is selected, and under it, the 'Tables' node is expanded, showing 'Customers', 'Customers\_Contact', 'Cust\_Surgery', 'DR\_Surgeries', 'Products', 'Product\_Specs', and 'Stock'. The main pane displays the following SQL code:

```
296
297 #Creating a query to select all customers registered to a certain doctors surgery
298 #This would be a useful query if there were ever an issue with a doctor's surgery, for example delayed repeat prescriptions,
299 #so that the pharmacy could see which customers would be effected.
300 • select Customer_ID
301   from Cust_Surgery
302   where Surgery_ID = 'S3'
303   order by Surgery_ID;
304
305
```

Below the code is a 'Result Grid' table with a single column 'Customer\_ID'. The data is as follows:

Customer_ID
C1
C6

## Query 5 code and output:

The screenshot shows a database interface with a sidebar containing a tree view of schema objects. The 'Pharmacy' schema is selected, and under it, the 'Tables' node is expanded, showing 'Customers', 'Customers\_Contact', 'Cust\_Surgery', 'DR\_Surgeries', 'Products', and 'Product\_Specs'. The main pane displays the following SQL code:

```
304
305 #Creating a query to retrieve all products that start with the letter 'A':
306 #This could be useful if a customer is looking for a product but can't quite remember the full name, it allows pharmacy staff
307 #to search the product name likeness to see the possible options in the phamacy product stock.
308 • Select Product_Name
309   from Products
310   where Product_Name like 'A%'
311   order by Product_Name;
312
313 --- -----Using a query to delete data: -----
314
```

Below the code is a 'Result Grid' table with a single column 'Product\_Name'. The data is as follows:

Product_Name
Amoxicillin
Atenolol
Azithromycin

## Using a query to delete data:

```

> Country
> bank
> bookshop
> customers
> Danni_Birthday
> Jewellery_Shop
> Parts
> Pharmacy
  > Tables
    > Customers
    > Customers_Contact
    > Cust_Surgery
    > DR_Surgeries
    > Products
    > Product_Specs
    > Stock
313  -- -----Using a query to delete data: -----
314  #I am first going to create an extra row for the purpose of this query, as I don't want to delete my original dataset.
315  #So I am using an insert query to insert a new customer that I will use for the delete query:
316  • insert into Customers(
317    Customer_ID, First_Name, Surname, DOB, Known_Conditions)
318    values
319    ('C13', 'Peter', 'Parker', '1946-08-10', 'Penicillin Allergy');
320  #I am now using the select * from Customers to make sure this new row has been added.
321  • select * from Customers;
322
323  #I am now creating a delete query to delete this row of data from the table 'Customers' in the db.
324  • delete from Customers #Using the 'delete' keyword with 'from' to call the delete action and to indicate from which table.
325  where Customer_ID = 'C13'; #Using 'where' constraint to specify the exact part of the table to be deleted in this query.
326
327  #I am now using select * from Customers again to make sure this row has been deleted.
328  • select * from Customers;
329

```

## Output before delete:

	Customer_ID	First_Name	Surname	DOB	Known_Conditions
▶	C1	Rachel	Greene	1968-08-02	T1 Diabetes
◀	C10	Jean-Luc	Picard	1940-05-02	Anemia
◀	C11	Beverly	Crusher	1952-09-07	Depression
◀	C12	Geordi	Laforgue	1951-03-10	Glaucoma
◀	C13	Peter	Parker	1946-08-10	Penicillin Allergy
◀	C2	Monoica	Bing	1968-12-03	OCD
◀	C3	Chandler	Bing	1967-04-09	Eczema
◀	C4	Joey	Tribbiani	1965-01-07	T2 Diabetes
◀	C5	Ross	Geller	1962-06-09	Asthma
◀	C6	Pheobe	Buffay	1969-07-08	Anxiety
◀	C7	Gunther	Smith	1963-10-04	hypertension
◀	C8	Deanna	Troi	1958-11-25	Epilepsy
◀	C9	William	Ricker	1955-01-31	Arthritis
	HULL	HULL	HULL	HULL	HULL

## Output after delete:

	Customer_ID	First_Name	Surname	DOB	Known_Conditions
▶	C1	Rachel	Greene	1968-08-02	T1 Diabetes
◀	C10	Jean-Luc	Picard	1940-05-02	Anemia
◀	C11	Beverly	Crusher	1952-09-07	Depression
◀	C12	Geordi	Laforgue	1951-03-10	Glaucoma
◀	C2	Monoica	Bing	1968-12-03	OCD
◀	C3	Chandler	Bing	1967-04-09	Eczema
◀	C4	Joey	Tribbiani	1965-01-07	T2 Diabetes
◀	C5	Ross	Geller	1962-06-09	Asthma
◀	C6	Pheobe	Buffay	1969-07-08	Anxiety
◀	C7	Gunther	Smith	1963-10-04	hypertension
◀	C8	Deanna	Troi	1958-11-25	Epilepsy
◀	C9	William	Ricker	1955-01-31	Arthritis
	HULL	HULL	HULL	HULL	HULL

## Using a drop query to delete data:

```

> bookshop
> customers
> Danni_Birthday
> Jewellery_Shop
> Parts
> Pharmacy
  > Tables
    > Customers
    > Customers_Contact
    > Cust_Surgery
    > DR_Surgeries
    > Products
    > Product_Specs
    > Stock
    > Surgery_Info
330  -- using a drop query to delete data:
331  #I am going to create a quick 'temporary' table with values to demonstrate a drop table query.
332  #(As I don't want to delete any of the tables that I have created that relate to procedures views etc).
333  #example table creation:
334  • create table temporarytable(
335    Employee_ID int not null auto_increment primary key, First_Name varchar(50) not null, Surname varchar (50) not null, Title varchar(50) not
336    );
337  • insert into temporarytable (
338    First_Name, Surname, Title)
339    values
340    ('Kathryn', 'Janeway', 'Pharmacy Director'),
341    ('Annika', 'Hansen', 'Chemist'),
342    ('Tom', 'Paris', 'Sales Assistant');
343
344  #example drop query:
345  • drop table temporarytable;
346

```

## Output before query:

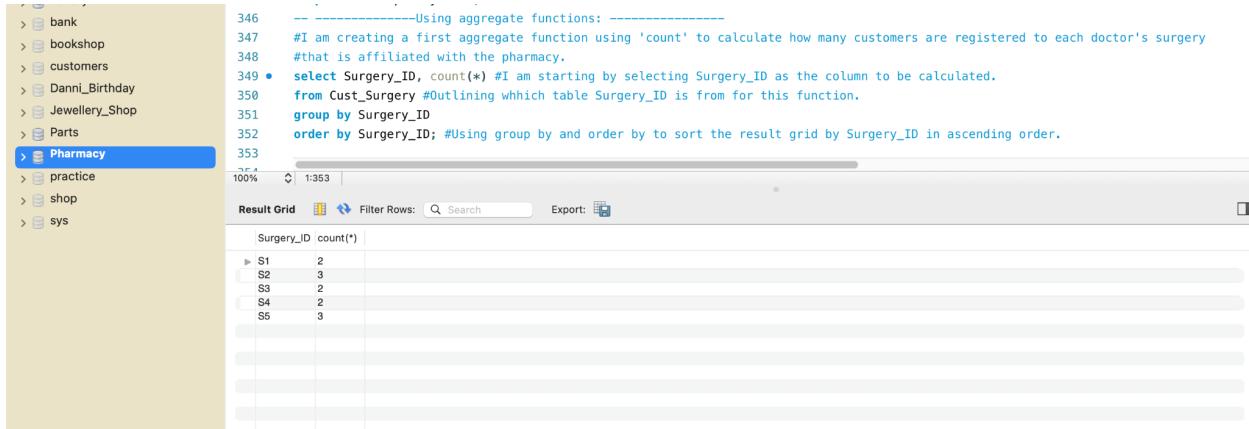
The screenshot shows a database navigation interface with a tree view. The root node is 'Pharmacy', which has a database icon. Below it is a 'Tables' node, indicated by a folder icon. Under 'Tables', there are nine entries, each preceded by a table icon: 'Customers', 'Customers\_Contact', 'Cust\_Surgery', 'DR\_Surgeries', 'Products', 'Product\_Specs', 'Stock', 'Surgery\_Info', and 'temporarytable'. The entire tree is contained within a light beige rectangular area.

## Output after query:

The screenshot shows a database navigation interface with a tree view. The root node is 'Pharmacy', which has a database icon. Below it is a 'Tables' node, indicated by a folder icon. Under 'Tables', there are nine entries, each preceded by a table icon: 'Customers', 'Customers\_Contact', 'Cust\_Surgery', 'DR\_Surgeries', 'Products', 'Product\_Specs', 'Stock', 'Surgery\_Info', and 'temporarytable'. To the right of the 'Surgery\_Info' entry, there are three small circular icons with symbols: a blue circle with a white 'i', a green circle with a white question mark, and a red circle with a white exclamation mark. Below the 'Tables' node, there is a new node 'Views', indicated by a folder icon. The entire tree is contained within a light beige rectangular area.

## Aggregate Functions:

### Aggregate function 1 and output:



The screenshot shows a database interface with a sidebar containing various schema names like bank, bookshop, customers, Danni\_Birthday, Jewellery\_Shop, Parts, and Pharmacy. The Pharmacy schema is selected and highlighted in blue. The main area displays a SQL query and its execution results.

```

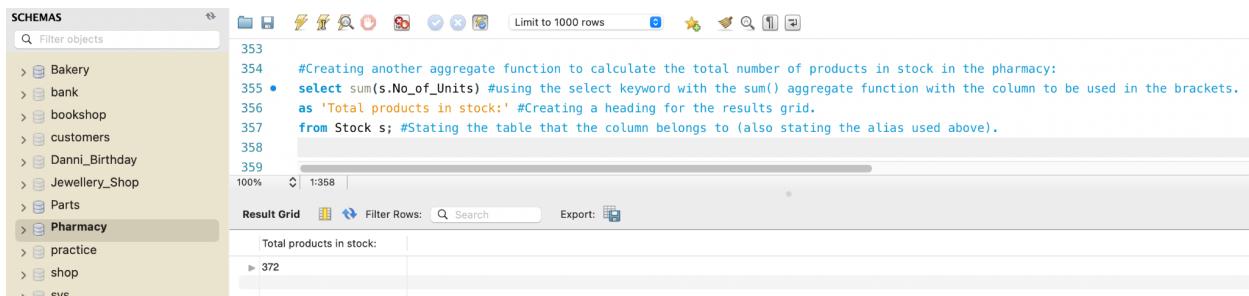
346  -- -----Using aggregate functions: -----
347  #I am creating a first aggregate function using 'count' to calculate how many customers are registered to each doctor's surgery
348  #that is affiliated with the pharmacy.
349 • select Surgery_ID, count(*) #I am starting by selecting Surgery_ID as the column to be calculated.
350  from Cust_Surgery #Outlining which table Surgery_ID is from for this function.
351  group by Surgery_ID
352  order by Surgery_ID; #Using group by and order by to sort the result grid by Surgery_ID in ascending order.
353

```

The Result Grid shows the following data:

Surgery_ID	count(*)
S1	2
S2	3
S3	2
S4	2
S5	3

### Aggregate function 2 and output:



The screenshot shows a database interface with a sidebar containing various schema names like Bakery, bank, bookshop, customers, Danni\_Birthday, Jewellery\_Shop, Parts, and Pharmacy. The Pharmacy schema is selected and highlighted in blue. The main area displays a SQL query and its execution results.

```

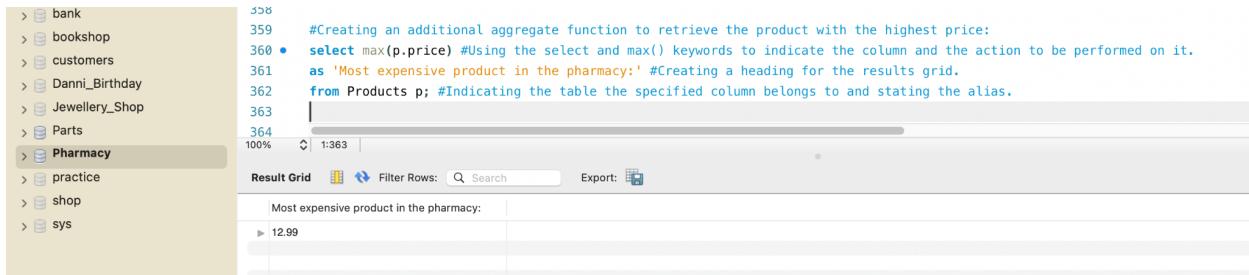
353
354  #Creating another aggregate function to calculate the total number of products in stock in the pharmacy:
355 • select sum(s.No_of_Units) #using the select keyword with the sum() aggregate function with the column to be used in the brackets.
356  as 'Total products in stock:' #Creating a heading for the results grid.
357  from Stock s; #Stating the table that the column belongs to (also stating the alias used above).
358
359

```

The Result Grid shows the following data:

Total products in stock:
372

### Aggregate function 3 and output:



The screenshot shows a database interface with a sidebar containing various schema names like bank, bookshop, customers, Danni\_Birthday, Jewellery\_Shop, Parts, and Pharmacy. The Pharmacy schema is selected and highlighted in blue. The main area displays a SQL query and its execution results.

```

358
359  #Creating an additional aggregate function to retrieve the product with the highest price:
360 • select max(p.price) #Using the select and max() keywords to indicate the column and the action to be performed on it.
361  as 'Most expensive product in the pharmacy:' #Creating a heading for the results grid.
362  from Products p; #Indicating the table the specified column belongs to and stating the alias.
363
364

```

The Result Grid shows the following data:

Most expensive product in the pharmacy:
12.99

## Aggregate function 4 and output:

The screenshot shows a database interface with a sidebar titled 'SCHEMAS' containing various database objects like Bakery, bank, bookshop, customers, Danni\_Birthday, Jewellery\_Shop, Parts, and Pharmacy. The Pharmacy schema is selected. The main area displays a SQL query and its results.

```
363 #Using an aggregate function to calculate the difference in price of the most expensive and least expensive products in stock
364 #at the pharmacy:
365
366 • select round(max(p.price) - min(p.price), 2) # Using select and max() keywords to state the action and the column to work on.
367 #I have also used round() with figure 2 to round the answer to a decimal place (and so like a monetary value).
368 as 'Price Difference:' #Creating a heading for the result grid.
369 from Products p; #Stating the table the column belongs to and the alias used.
370
371
```

Result Grid

Price Difference...
12.6

## Using Joins:

### Right join and output:

The screenshot shows a database interface with a sidebar titled 'SCHEMAS' containing various database objects like Bakery, bank, bookshop, customers, Danni\_Birthday, Jewellery\_Shop, Parts, and Pharmacy. The Pharmacy schema is selected. The main area displays a SQL query for a right join.

```
372 -----Using joins:
373 #I am creating a right join to join together the customers' first and last names with their phone numbers.
374 #This would be useful for the pharmacy if they needed to contact a customer for a time-sensitive matter.
375 • select Customers.First_Name, Customers.Surname, Customers_Contact.Phone #Outlining the columns for the join.
376 from Customers #stating the first table for the join
377 right join Customers_Contact #stating the type of join (here right join) and the second table to join with,
378 on Customers.Customer_ID = Customers_Contact.Customer_ID #Aligning the data.
379 order by Customers.First_Name; #Using order by to sort the data in the join by ascending customer first name.
380
```

Result Grid

First_Name	Surname	Phone
Beverly	Crusher	07959632381
Chandler	Bing	07685943491
Deanna	Troi	07858654568
Geordi	Lafarge	07545859626
Gunther	Smith	07656392914
Jean-Luc	Picard	07652858965
Joey	Tribbiani	07848210365
Monica	Bing	07865485973
Pheobe	Buffay	07875262895
Rachel	Greene	02035079861
Ross	Geller	07986525326
William	Ricker	07646285393

## Left join and output:

The screenshot shows a database interface with a sidebar containing schema names like Parts, Pharmacy, practice, shop, and sys. The Pharmacy schema is selected. The main area displays a SQL query and its results. The query is:

```
381 #Creating a second join, this time a left join.  
382  
383 • select Product_Specs.Product_ID, Product_Specs.Manufacturer, Stock.DeliveryTime_Days #outlining the columns for the join.  
384 from Product_Specs #Stating the first table for the join  
385 left join Stock #stating the type of join (here left join) and the second table to combine the first with.  
386 on Product_Specs.Product_ID = Stock.Product_ID #Aligning the different columns.  
387 order by Product_Specs.Manufacturer; #Sorting the data in the join by Manufacturer ascending order.  
388
```

The results grid shows the following data:

Product_ID	Manufacturer	DeliveryTime_Days
P11	Amarox Ltd	3
P15	Amarox Ltd	3
P4	Amarox Ltd	2
P9	Amarox Ltd	2
P2	Aspar Pharma	3
P3	Aspar Pharma	3
P13	Bayer Pharma...	6
P10	GlaxoSmithKline	2
P14	GlaxoSmithKline	4
P5	GlaxoSmithKline	4
P6	GlaxoSmithKline	3
P7	GlaxoSmithKline	5
P8	Karo Healthcare	4
P1	Novo-Nordisk	5
P12	Novo-Nordisk	5

On the right side of the interface, there are several icons for Result Grid, Form Editor, Field Types, and Query Stats.

## In-built functions:

### Date\_format function and output:

The screenshot shows a database interface with a sidebar containing schema names like Bakery, bank, bookshop, customers, Danni\_Birthday, Jewellery\_Shop, Parts, and sys. The Pharmacy schema is selected. The main area displays a SQL query and its results. The query is:

```
389 -----Using in-built functions: -----  
390  
391 #For age restrictions on medications, as well as dosages, it can be useful for the pharmacy to know the age of the customers.  
392 #I'm using the date_format inbuilt function to have a list of customer_IDs and a more readable format of their date of birth.  
393 #This should make it easier for pharmacists to see customer's age if need be.  
394 • select Customer_ID, date_format(DOB, '%e/%m/%y') #using the inbuilt function and selecting the column it effects.  
395 as 'Customer Formatted Date of Birth' #creating a heading to display the result under.  
396 from pharmacy.Customers;  
397
```

The results grid shows the following data:

Customer_ID	Customer Formatted Date of Birth
C1	2/08/68
C10	2/05/40
C11	7/09/52
C12	10/03/51
C2	3/12/68
C3	9/04/67
C4	7/01/65
C5	9/06/62
C6	8/07/69
C7	4/10/63
C8	25/11/58
C9	31/01/55

On the right side of the interface, there are several icons for Result Grid, Form Editor, Field Types, and Query Stats.

## Ceiling function and output:

The screenshot shows a database interface with a sidebar titled 'SCHEMAS' containing tables like Bakery, bank, bookshop, customers, Danni\_Birthday, Jewellery\_Shop, Parts, and Pharmacy. The 'Pharmacy' table is selected and highlighted in grey. The main area displays a SQL query and its results.

```
397
398     #For accounting purposes it might be useful to know the lowest rounded amount of each price (for future discounts or something).
399     #So here I am using the ceiling() in-built function to round the price but not be less than the value of the price.
400 •   select ceiling(Price) #using the inbuilt function and selecting the column it effects.
401     as 'Rounded Prices of Products' #creating a heading to display the result under.
402     from pharmacy.Products; #indicating the table to get the column from.
403
404     #With the repeat prescriptions it can be useful to know when the next one will be due so the pharmacy knows if they need to
405     #reorder stock and with how much time etc. So using the the adddate function would allow the pharmacists to enter the date
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
```

The results grid shows a single column labeled 'Rounded Prices of Products' with values ranging from 8 to 10.

## Adddate function and output:

The screenshot shows a database interface with a sidebar titled 'SCHEMAS' containing tables like bank, bookshop, customers, Danni\_Birthday, Jewellery\_Shop, Parts, and Pharmacy. The 'Pharmacy' table is selected and highlighted in grey. The main area displays a SQL query and its results.

```
403
404     #With the repeat prescriptions it can be useful to know when the next one will be due so the pharmacy knows if they need to
405     #reorder stock and with how much time etc. So using the the adddate function would allow the pharmacists to enter the date
406     #they fill a prescription, and then it can automatically be calculated when the next one will be due.
407 •   select adddate('2023-10-01', interval 21 day) #using the inbuilt function and selecting the column it effects.
408     as 'Next prescription due'; #creating a heading to display the result under.
409
410
411
412
413
414
415
416
417
418
419
420
421
```

The results grid shows a single column labeled 'Next prescription due' with the value '2023-10-22'.

## Creating a View:

### View code:

The screenshot shows a database interface with a sidebar titled 'SCHEMAS' containing tables like Bakery, bank, bookshop, customers, Danni\_Birthday, Jewellery\_Shop, Parts, and Pharmacy. The 'Pharmacy' table is selected and highlighted in grey. The main area displays a SQL query for creating a view.

```
409
410     -- ----- Creating a view -----
411     #I would like to create a view of each product name, their manufacturer and the delivery time to order more units into the pharmacy.
412     #This view would be useful for the pharmacy because when it comes to re-ordering stock, all information is in one place,
413     #rather than across 3 different tables.
414 •   create view Pdt_MFR_Dvry as
415     select Product_Name, Manufacturer, DeliveryTime_Days
416     from Products
417     inner join Product_Specs #I am using an join (here inner join) to create the view as it involves 3 tables.
418     on Products.Product_ID = Product_Specs.Product_ID
419     inner join Stock
420     on Product_Specs.Product_ID = Stock.Product_ID;
421
```

## View output:

The screenshot shows a database interface with a sidebar containing a tree view of schemas and tables. The 'Pharmacy' schema is selected. A query window displays the results of the following SQL statement:

```
1 •  SELECT * FROM Pharmacy.pdt_mfr_dvry;
```

The results are presented in a grid format:

Product_Name	Manufacturer	DeliveryTime_Days
Novorapid Insulin	Novo-Nordisk	5
Atenolol	GlaxoSmithKline	2
Dapoxetine	Amarox Ltd	3
Levemir Insulin	Novo-Nordisk	5
Canesten Cream	Bayer Pharmaceuticals	6
Carvedilol	GlaxoSmithKline	4
Citalopram	Amarox Ltd	3
Paracetamol	Aspar Pharma	3
Sertraline	Amarox Ltd	2
Amoxicillin	GlaxoSmithKline	4
Cephalexin	GlaxoSmithKline	3
Azithromycin	GlaxoSmithKline	5
Lamisil	Karo Healthcare	4
Piritize	Amarox Ltd	2

## Stored Procedures:

### Procedure 1 code:

The screenshot shows a database interface with a sidebar containing a tree view of schemas and tables. The 'Pharmacy' schema is selected. A code editor window displays the creation of a stored procedure:

```
-- -----Creating stored procedures to achieve a goal: -----
#I am creating a stored procedure to find general product information from the product ID in the pharmacy.

delimiter $$ #Changing the delimiter from the ';' sign to be able to repeatedly use ';' in my code without ending the procedure.
create procedure find_product (in id varchar(25)) #Naming the procedure and defining the data to be entered to be searched.
begin #Starting the procedure body of code.
select * from Products #Outlining the information for the procedure to retrieve.
where Product_ID = id; #Matching the data entered by the user to the cell in the table to retrieve the data specific to it.
end $$ #marking the end of the procedure code.
delimiter ; #Returning the delimiter back to ';' to continue coding normally.
```

### Calling Procedure 1:

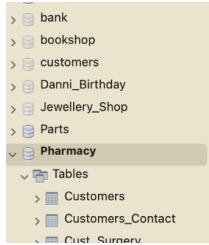
The screenshot shows a database interface with a sidebar containing a tree view of schemas and tables. The 'Pharmacy' schema is selected. A query window displays the execution of the stored procedure:

```
446 •  -- -----Using the stored procedures to achieve a goal: -----
447 call find_product('P3'); #Calling the stored procedure to find out basic information on product 'P3'.
448
```

The results are presented in a grid format:

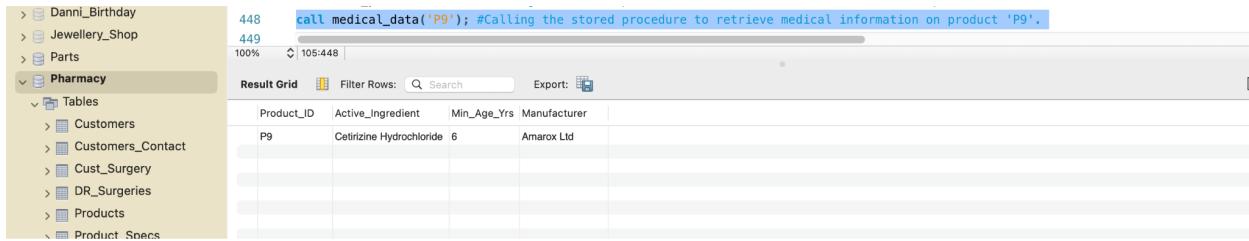
Product_ID	Product_Name	Product_Type	Price
P3	Ibuprofen	NSAID	0.4

## Procedure 2 code:



```
-- Creating a second stored procedure that will allow pharmacists to query medical information about a product
# from the product id. This will make it easier and faster for the pharmacy to retrieve this important information quickly.
#By having this as a stored procedure it also means the pharmacy can repeat this query over and over again by calling it.
delimiter $$ #Changing the delimiter from the ';' sign to be able to repeatedly use ';' in my code without ending the procedure.
create procedure medical_data (in id varchar(25)) #Naming the procedure and defining the data to be entered to be searched.
begin #starting the procedure code.
select * from Product_Specs #Outlining the information for the procedure to retrieve.
where Product_ID = id; #Matching the data entered by the user to the cell in the table to retrieve the data specific to it.
end $$ #marking the end of the procedure code.
delimiter ; #Setting the delimiter back to it's original value.
```

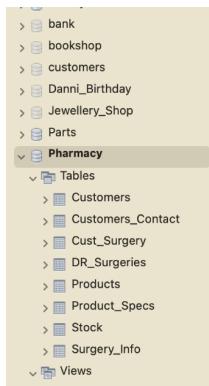
## Calling Procedure 2:



```
call medical_data('P9'); #Calling the stored procedure to retrieve medical information on product 'P9'.
```

Product_ID	Active_Ingredient	Min_Age_Yrs	Manufacturer
P9	Cetirizine Hydrochloride	6	AmaroX Ltd

## Updating 10 rows of mock data with dml command:



```
-- -----Updating data with 10 DML commands as per requirements: -----
#I have already used insert DML command to populate the db tables, and have used delete for delete query.
#I have also used select for queries.
#Please see here demonstrating the 'update' dml command:
SET SQL_SAFE_UPDATES = 0; #Turning off safe mode so that we can update without an error.
#The pharmacy would routinely need to alter the stock levels, here is an example of how:
update Stock set No_of_Units = 11 where Product_ID = 'P1'; #Update key word, set to update info and where to specify the row.
#It is also likely that a customer's contact information would change, so here is an example of how:
update Customers_Contact set Phone = '07986523526' where Customer_ID = 'C3'; #Update keyword, set updates & where specifies row.
update Products set Price = 3.12 where Product_ID = 'P9';
update Customers set Known_Conditions = 'Liver Disease' where Customer_ID = 'C10';
update Surgery_Info set Phone = '0205939648' where Surgery_ID = 'S1';
update Product_Specs set Manufacturer = 'Bayer', where Product_ID = 'P2';
update Product_Specs set Min_Age_Years = 8 where Product_ID = 'P2';
update Surgery_Info set Email = 'prescriptions@FGSurgery.nhs.uk' where Surgery_ID = 'S5';
update Products set Product_Name = 'Atenolol AN' where Product_ID = 'P10';
update Stock set DeliveryTime_Days = 1 where Product_ID = 'P3';

#--- Thanks! :)-----#
```