THE RELATIONAL DATABASE DESIGN/MODELING/IMPLEMENTATION IN SQL BASED SYSTEM ORACLE AND A RDBMS SOFTWARE MYSQL

JAYCE JONES

FALL 2018
CS4332

SUBMITTED TO PROFFESOR:
DR. C.J. HWANG

DECMBER 2, 2018

TEXAS STATE UNIVERSITY

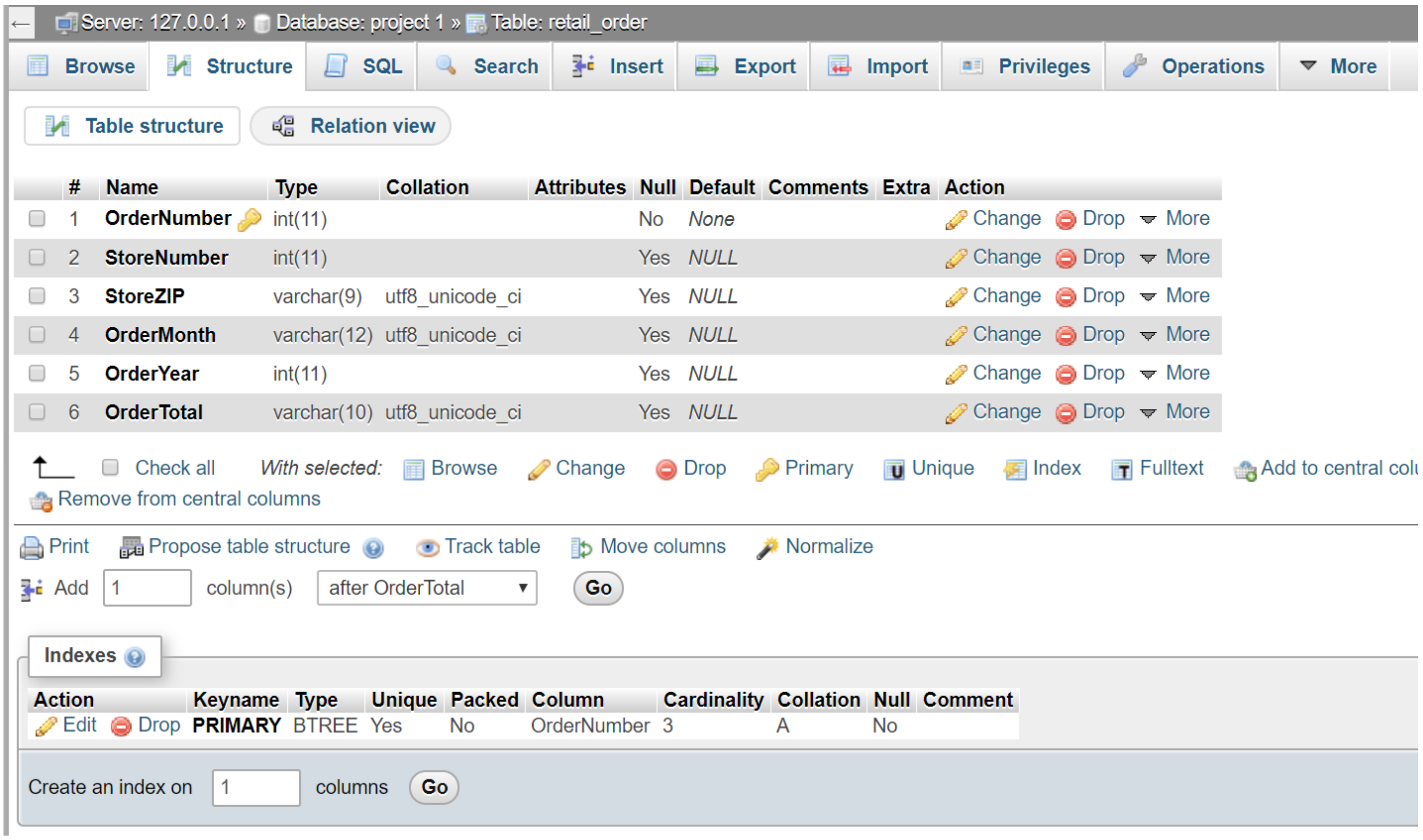
Table of Contents

Chapter 1 – Practicing MySQL

1.1.– Project Description

The goal of project 1 was to practice using MySQL. The project was completed using phpMyAdmin and MySQL. The project first involved accessing phpMyAdmin by using XAMPP Control Panel v3.2.2. This service created, and allowed for, access of a database on my local server where I was able to create, and store tables programmed in MySQL. To complete the project 3 tables were created using the table creation tool provided in PHP. The tables together represented a retail sales database. The first table created held data on a retail order, the data in the retail order table included: order number, store number, store zip, order month, order year, and order total. The seconds table created held data on an order item, the data in the retail order table included: order number, SKU, quantity, price, and extended price. The third table created held data on SKU's, the data in the SKU data table included: SKU, SKU description, department, and buyer. All data in the tables was entered using the PHP data entry tool. The project was successful in allowing one to familiarize oneself with the basics on my SQL and PHP.

1.2 – Tables/Input/Queries and Associated Output



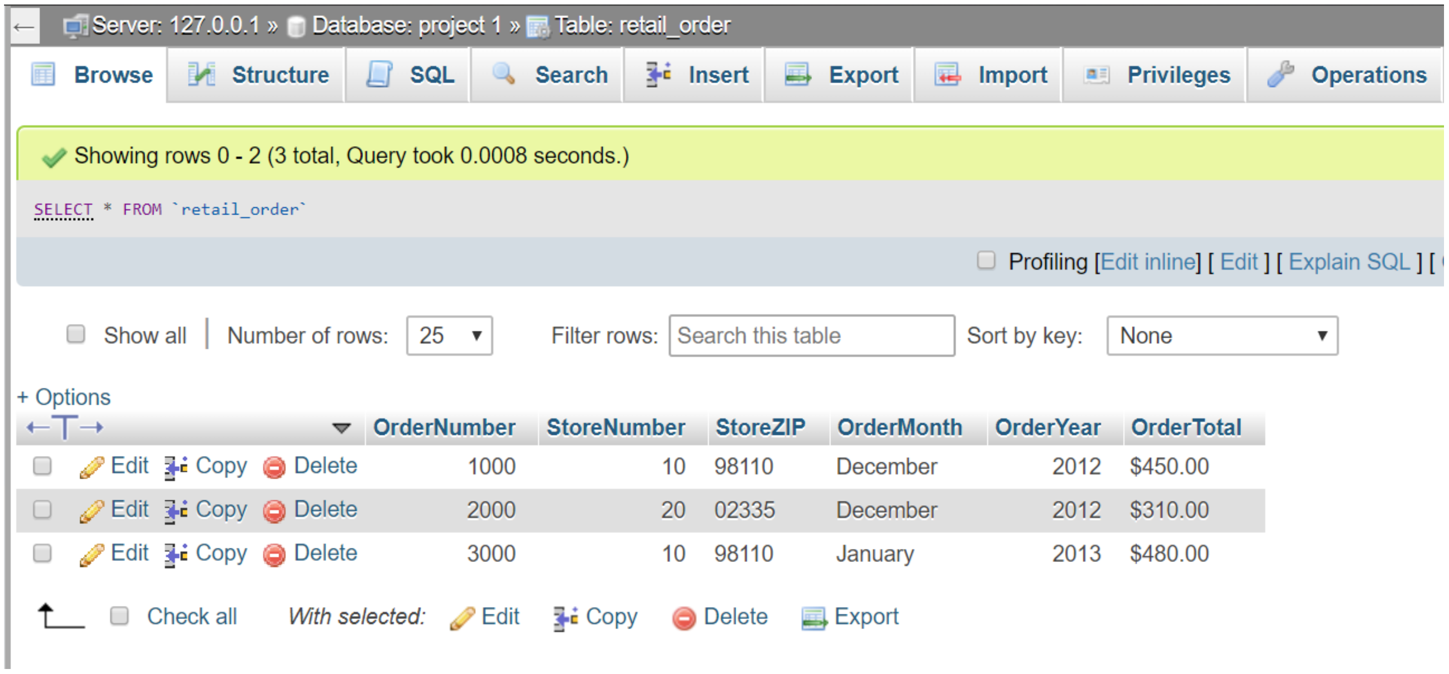
Server: 127.0.0.1 » Database: project 1 » Table: retail_order
Browse Structure SQL Search Insert Export Import Privileges Operations More
Table structure Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	OrderNumber	int(11)			No	None			Change Drop More
2	StoreNumber	int(11)			Yes	NULL			Change Drop More
3	StoreZIP	varchar(9)	utf8_unicode_ci		Yes	NULL			Change Drop More
4	OrderMonth	varchar(12)	utf8_unicode_ci		Yes	NULL			Change Drop More
5	OrderYear	int(11)			Yes	NULL			Change Drop More
6	OrderTotal	varchar(10)	utf8_unicode_ci		Yes	NULL			Change Drop More

Check all With selected: Browse Change Drop Primary Unique Index Fulltext Add to central colu
Remove from central columns
Print Propose table structure Track table Move columns Normalize
Add 1 column(s) after OrderTotal Go
Indexes

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Drop	PRIMARY	BTREE	Yes	No	OrderNumber	3	A	No	

Retail_Order Table Structure



Server: 127.0.0.1 » Database: project 1 » Table: retail_order
Browse Structure SQL Search Insert Export Import Privileges Operations

Showing rows 0 - 2 (3 total, Query took 0.0008 seconds.)

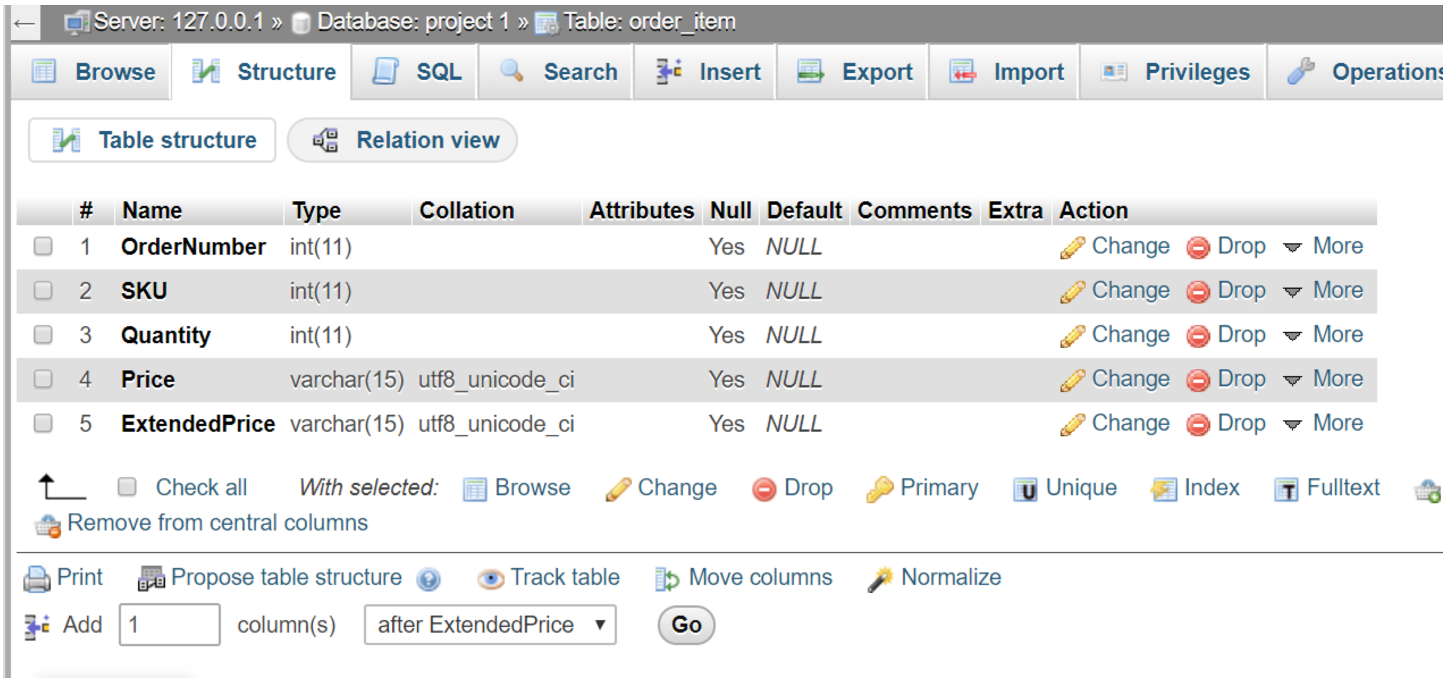
```
SELECT * FROM `retail_order`
```

Profiling [Edit inline] [Edit] [Explain SQL] [
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
+ Options

	OrderNumber	StoreNumber	StoreZIP	OrderMonth	OrderYear	OrderTotal
Edit Copy Delete	1000	10	98110	December	2012	$450.00
Edit Copy Delete	2000	20	02335	December	2012	$310.00
Edit Copy Delete	3000	10	98110	January	2013	$480.00

Check all With selected: Edit Copy Delete Export

Retail_Order Table


Server: 127.0.0.1 » Database: project 1 » Table: order_item

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations

Table structure | Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**OrderNumber**	int(11)			Yes	*NULL*			Change Drop More
2	**SKU**	int(11)			Yes	*NULL*			Change Drop More
3	**Quantity**	int(11)			Yes	*NULL*			Change Drop More
4	**Price**	varchar(15)	utf8_unicode_ci		Yes	*NULL*			Change Drop More
5	**ExtendedPrice**	varchar(15)	utf8_unicode_ci		Yes	*NULL*			Change Drop More

Check all *With selected:* Browse Change Drop Primary Unique Index Fulltext
Remove from central columns

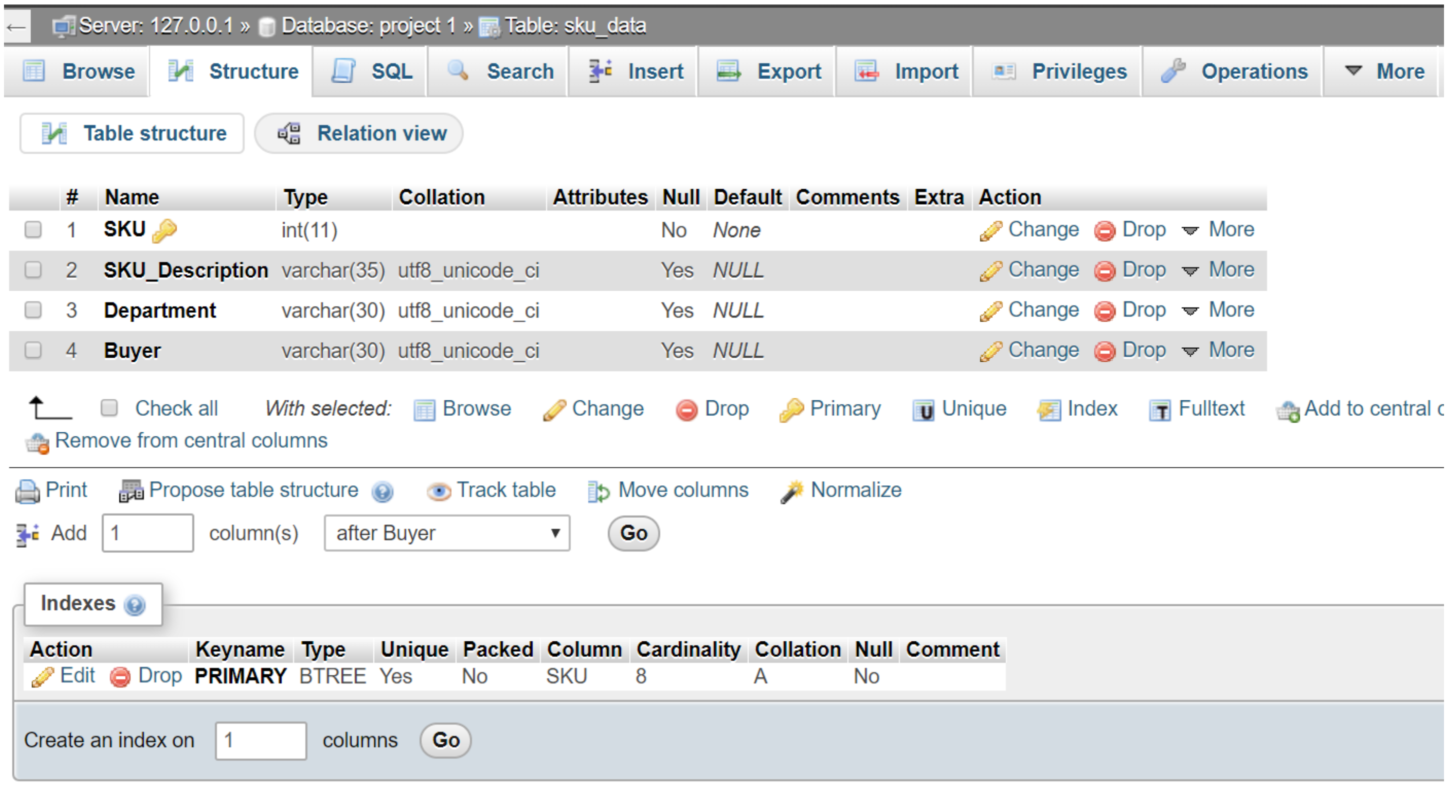
Print Propose table structure Track table Move columns Normalize

Add 1 column(s) after ExtendedPrice Go

Order_Item Table Structure

OrderNumber	SKU	Quantity	Price	ExtendedPrice
1000	201000	1	$300.00	$300.00
1000	202000	1	$130.00	$130.00
2000	101100	4	$50.00	$200.00
2000	101200	2	$50.00	$100.00
3000	100200	1	$300.00	$300.00
3000	101100	2	$50.00	$100.00
3000	101200	1	$50.00	$50.00

Order_Item Table



Server: 127.0.0.1 » Database: project 1 » Table: sku_data

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | More

Table structure | Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**SKU**	int(11)			No	*None*			Change Drop More
2	**SKU_Description**	varchar(35)	utf8_unicode_ci		Yes	*NULL*			Change Drop More
3	**Department**	varchar(30)	utf8_unicode_ci		Yes	*NULL*			Change Drop More
4	**Buyer**	varchar(30)	utf8_unicode_ci		Yes	*NULL*			Change Drop More

Check all *With selected:* Browse Change Drop Primary Unique Index Fulltext Add to central c
Remove from central columns

Print Propose table structure Track table Move columns Normalize

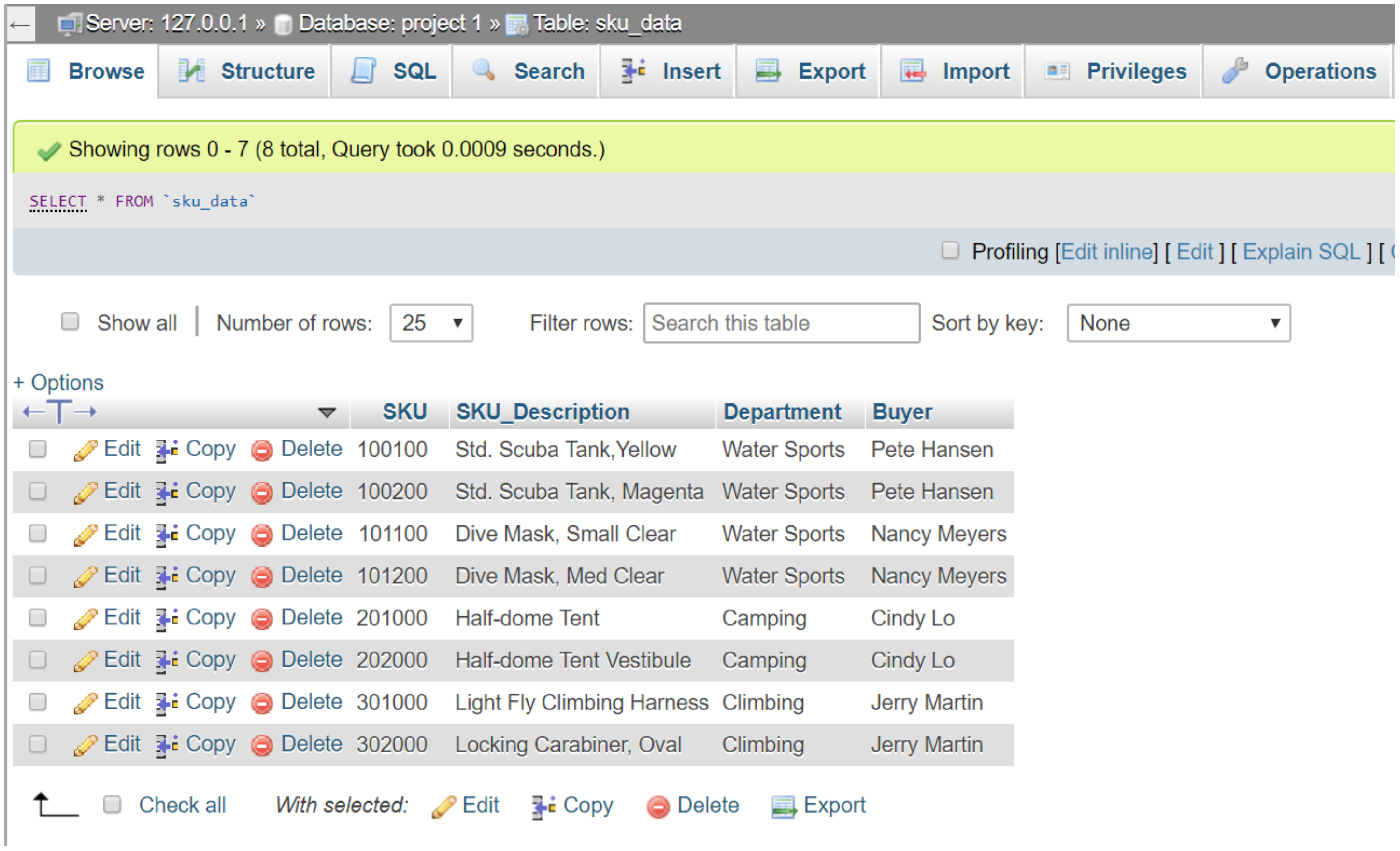
Add 1 column(s) after Buyer Go

Indexes

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Drop	**PRIMARY**	BTREE	Yes	No	SKU	8	A	No	

Create an index on 1 columns Go

SKU_Data Table Structure



Server: 127.0.0.1 » Database: project 1 » Table: sku_data

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations

Showing rows 0 - 7 (8 total, Query took 0.0009 seconds.)

```
SELECT * FROM `sku_data`
```

Profiling [Edit inline] [Edit] [Explain SQL] [

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

	SKU	SKU_Description	Department	Buyer
Edit Copy Delete	100100	Std. Scuba Tank,Yellow	Water Sports	Pete Hansen
Edit Copy Delete	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
Edit Copy Delete	101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers
Edit Copy Delete	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers
Edit Copy Delete	201000	Half-dome Tent	Camping	Cindy Lo
Edit Copy Delete	202000	Half-dome Tent Vestibule	Camping	Cindy Lo
Edit Copy Delete	301000	Light Fly Climbing Harness	Climbing	Jerry Martin
Edit Copy Delete	302000	Locking Carabiner, Oval	Climbing	Jerry Martin

Check all *With selected:* Edit Copy Delete Export

SKU_Data Table

Chapter 2 – Relations and Queries

2.1. Project Description

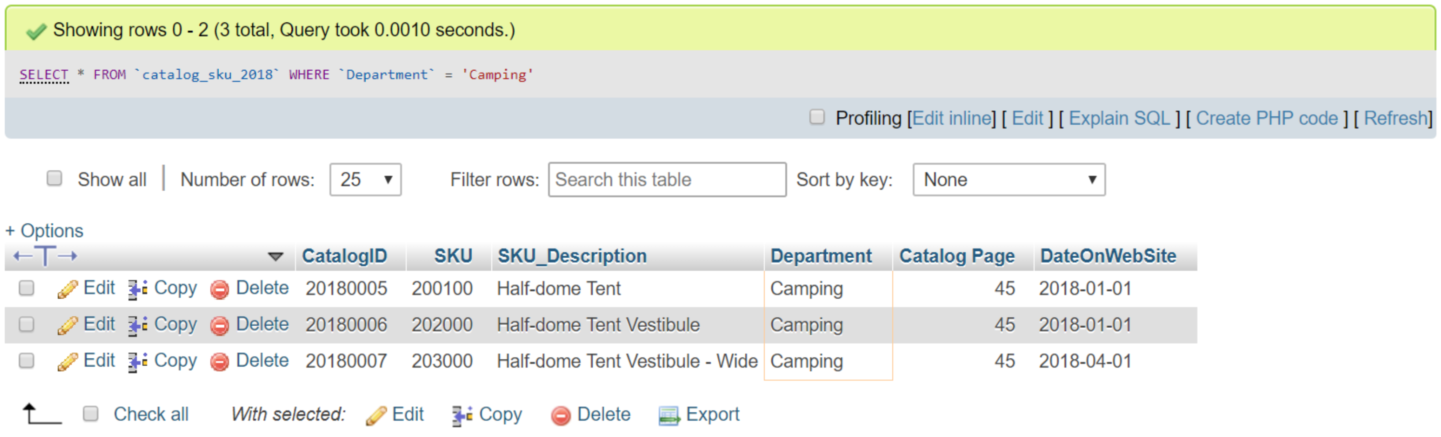
The goal of project 2 was to create relations among tables that would allow for queries to be performed on the database. The project was completed using phpMyAdmin and MySQL. The tables used in this project were those already created in project 1 along with 3 other tables created in the same way as the tables in project 1 .In order to perform queries on the tables relations among the tables first has to be created. Relations were created by linking the tables using primary and foreign keys. For the retail order table, the primary key was assigned to order number. For the SKU data table, the primary key was assigned to the SKU. For the buyer table, the primary key was assigned to the buyer name. and in the catalog SKU yearly tables the primary key was assigned to the catalog ID. In the order item table order number and SKU acted as foreign keys, linking all tables together and allowing for queries to be carried out on the table data. With the tables linked, 8 non-trivial SQL queries were carried out on the database. SQL statements were written to query the tables, along with an English statement to explain each query. These statements along with the resulting output may be seen in *2.2.* below. An E/R model was also created which allows for one to visually see the relations among tables. The project was successful in allowing one to become familiar with relations among tables in a database and to develop an understanding of how basic queries are written to return desired data.

2.2. Tables/Input/Queries and Associated Output

Query 1: Where Equal

A query that returns all rows from the Catalog_SKU_2018 table where the Department name <u>is equal to</u> 'Camping'.

```
SELECT *
FROM `catalog_sku_2018`
WHERE `Department` = 'Camping';
```



Showing rows 0 - 2 (3 total, Query took 0.0010 seconds.)

```
SELECT * FROM `catalog_sku_2018` WHERE `Department` = 'Camping'
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

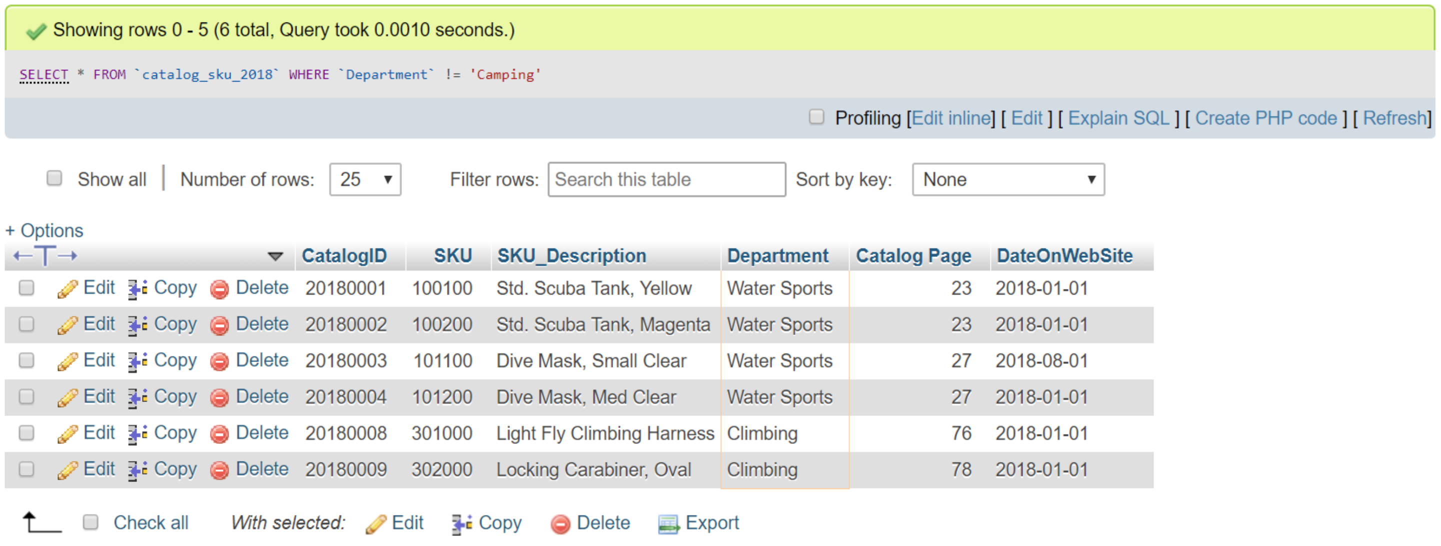
CatalogID	SKU	SKU_Description	Department	Catalog Page	DateOnWebSite
20180005	200100	Half-dome Tent	Camping	45	2018-01-01
20180006	202000	Half-dome Tent Vestibule	Camping	45	2018-01-01
20180007	203000	Half-dome Tent Vestibule - Wide	Camping	45	2018-04-01

Check all With selected: Edit Copy Delete Export

Query 2: Where Not Equal

A query that returns all rows from the Catalog_SKU_2018 table where the Department name <u>is not equal to</u> 'Camping'.

```
SELECT *
FROM `catalog_sku_2018`
WHERE `Department` != 'Camping';
```



Showing rows 0 - 5 (6 total, Query took 0.0010 seconds.)

```
SELECT * FROM `catalog_sku_2018` WHERE `Department` != 'Camping'
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

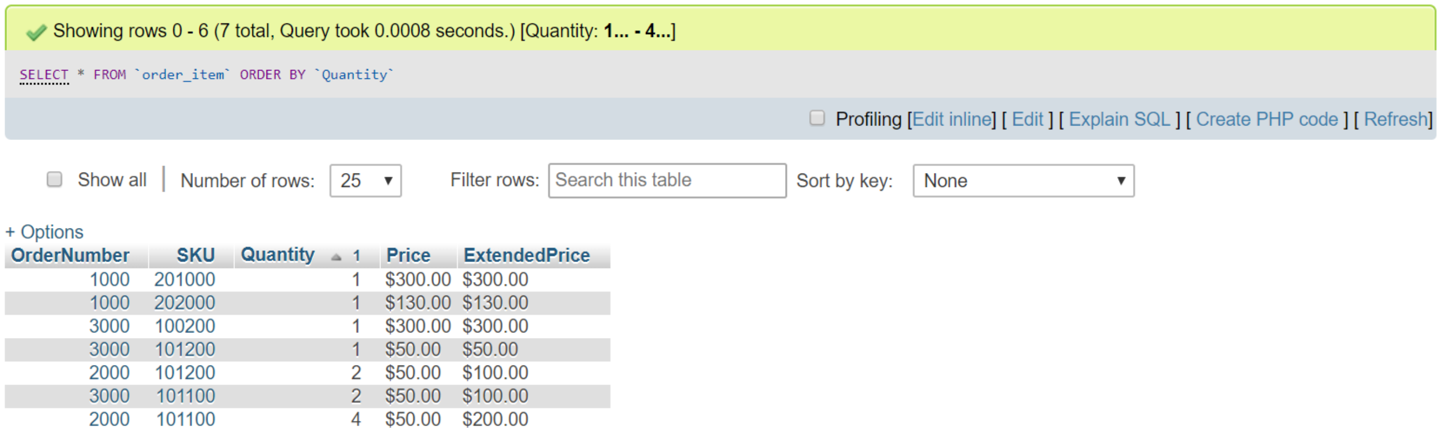
CatalogID	SKU	SKU_Description	Department	Catalog Page	DateOnWebSite
20180001	100100	Std. Scuba Tank, Yellow	Water Sports	23	2018-01-01
20180002	100200	Std. Scuba Tank, Magenta	Water Sports	23	2018-01-01
20180003	101100	Dive Mask, Small Clear	Water Sports	27	2018-08-01
20180004	101200	Dive Mask, Med Clear	Water Sports	27	2018-01-01
20180008	301000	Light Fly Climbing Harness	Climbing	76	2018-01-01
20180009	302000	Locking Carabiner, Oval	Climbing	78	2018-01-01

Check all With selected: Edit Copy Delete Export

Query 3: Order

A query that returns all rows from the order_item table and orders them by Quantity.

```
SELECT *
FROM `order_item`
ORDER BY `Quantity`;
```



Showing rows 0 - 6 (7 total, Query took 0.0008 seconds.) [Quantity: **1... - 4...**]

```
SELECT * FROM `order_item` ORDER BY `Quantity`
```

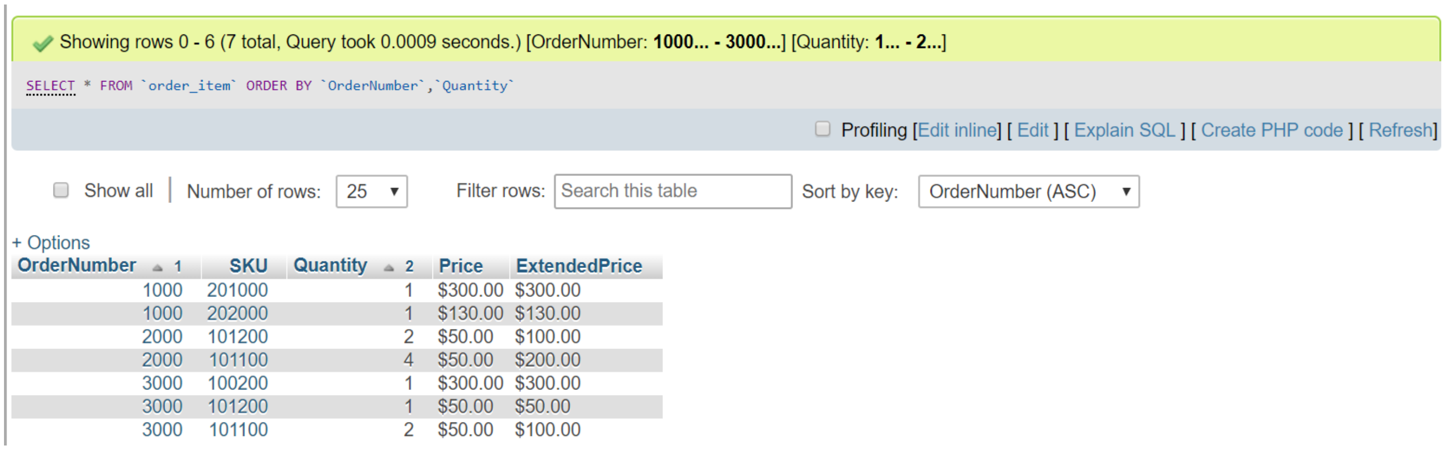
Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

OrderNumber	SKU	Quantity 1	Price	ExtendedPrice
1000	201000	1	$300.00	$300.00
1000	202000	1	$130.00	$130.00
3000	100200	1	$300.00	$300.00
3000	101200	1	$50.00	$50.00
2000	101200	2	$50.00	$100.00
3000	101100	2	$50.00	$100.00
2000	101100	4	$50.00	$200.00

Query 4: Order by Two Attributes

A query that returns all rows from the order_item table and orders them by OrderNumber and then by Quantity within the order number.

```
SELECT *
FROM `order_item`
ORDER BY `OrderNumber`,`Quantity`;
```



Showing rows 0 - 6 (7 total, Query took 0.0009 seconds.) [OrderNumber: **1000... - 3000...**] [Quantity: **1... - 2...**]

```
SELECT * FROM `order_item` ORDER BY `OrderNumber`,`Quantity`
```

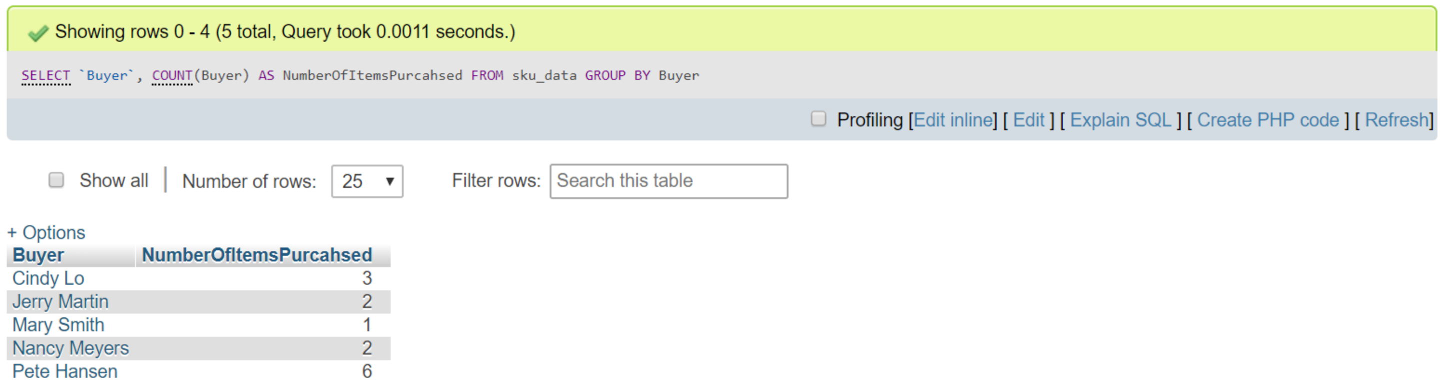
Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: OrderNumber (ASC)

OrderNumber 1	SKU	Quantity 2	Price	ExtendedPrice
1000	201000	1	$300.00	$300.00
1000	202000	1	$130.00	$130.00
2000	101200	2	$50.00	$100.00
2000	101100	4	$50.00	$200.00
3000	100200	1	$300.00	$300.00
3000	101200	1	$50.00	$50.00
3000	101100	2	$50.00	$100.00

Query 5: Group

A query on the sku_data table that returns a Buyer grouped with the number of items that they have purchased in a column called NumberOfItemsPurchased.

```
SELECT `Buyer`, COUNT(Buyer) AS NumberOfItemsPurcahsed
FROM sku_data
GROUP BY Buyer;
```



Showing rows 0 - 4 (5 total, Query took 0.0011 seconds.)

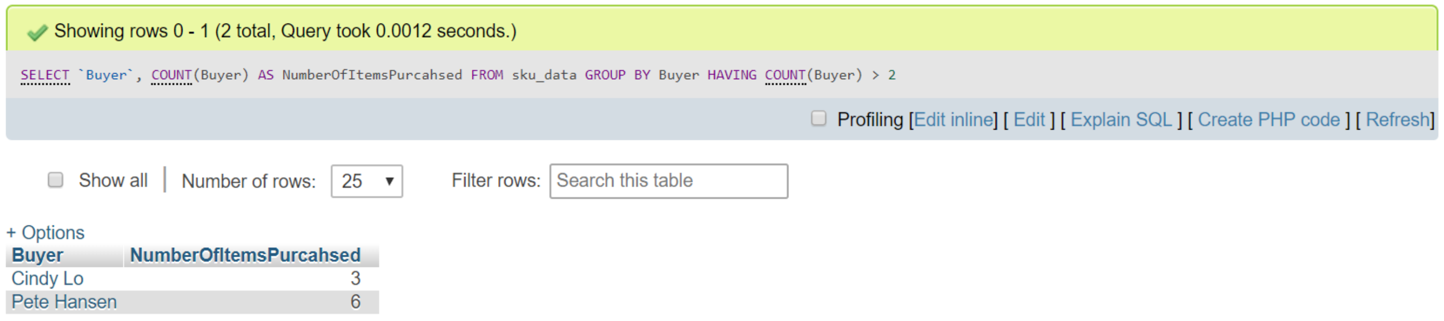
```
SELECT `Buyer`, COUNT(Buyer) AS NumberOfItemsPurcahsed FROM sku_data GROUP BY Buyer
```

Buyer	NumberOfItemsPurcahsed
Cindy Lo	3
Jerry Martin	2
Mary Smith	1
Nancy Meyers	2
Pete Hansen	6

Query 6: Group Having

A query on the sku_data table that returns a Buyer grouped with the number of items that they have purchased in a column called NumberOfItemsPurcahased if the Buyers have purchased more than 2 items.

```
SELECT `Buyer`, COUNT(Buyer) AS NumberOfItemsPurcahsed
FROM sku_data
GROUP BY Buyer
HAVING COUNT(Buyer) > 2;
```



Showing rows 0 - 1 (2 total, Query took 0.0012 seconds.)

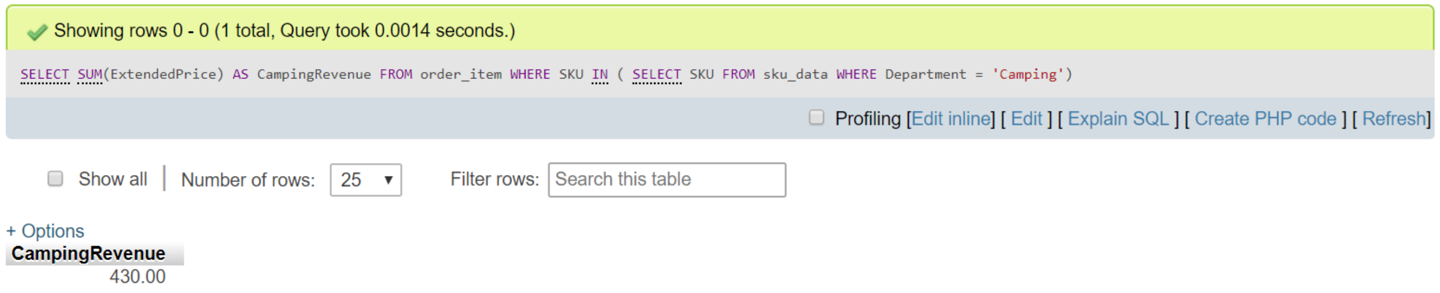
```
SELECT `Buyer`, COUNT(Buyer) AS NumberOfItemsPurcahsed FROM sku_data GROUP BY Buyer HAVING COUNT(Buyer) > 2
```

Buyer	NumberOfItemsPurcahsed
Cindy Lo	3
Pete Hansen	6

Query 7: Two Tables

A query to obtain the sum of the ExtendedPrice from the order_item table for items managed by the 'Camping' Department and allocate it to a column named CampingRevenue, the sum is obtained by completing a sub_query on the sku_table through SKU reference.

```
SELECT SUM(ExtendedPrice) AS CampingRevenue
FROM order_item
WHERE SKU IN
(
   SELECT SKU
   FROM sku_data
   WHERE Department = 'Camping');
```



```
Showing rows 0 - 0 (1 total, Query took 0.0014 seconds.)
SELECT SUM(ExtendedPrice) AS CampingRevenue FROM order_item WHERE SKU IN ( SELECT SKU FROM sku_data WHERE Department = 'Camping')
```

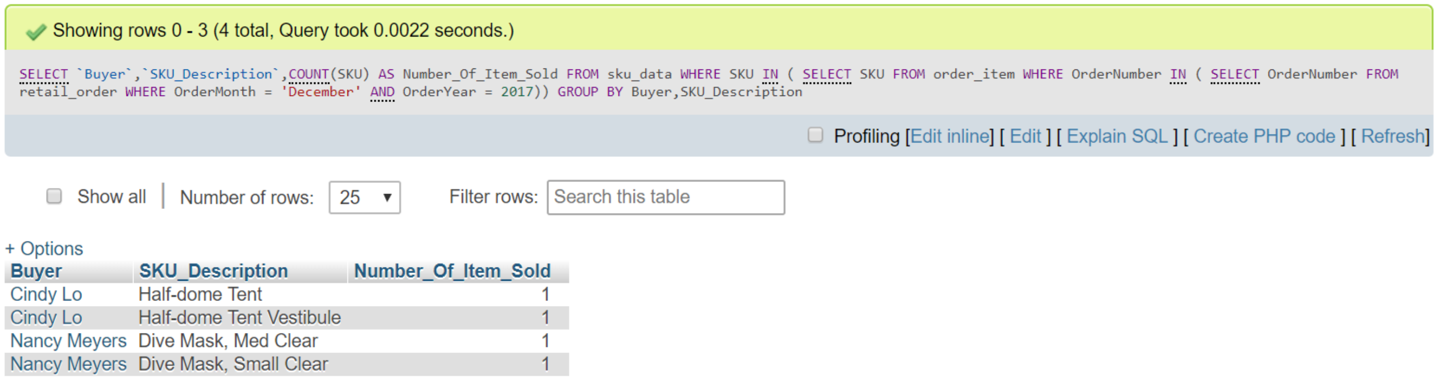
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25
Filter rows: Search this table
+ Options

CampingRevenue
430.00

Query 8: Two Tables by Grouping

A query to obtain the number of an item sold, grouped by the items SKU_Description and Buyers name. Obtained by querying the Buyer and SKU_Description from the sku_data table and sub_querying the order information from the retail_order table using the order_item table as a reference.

```
SELECT `Buyer`,`SKU_Description`,COUNT(SKU) AS Number_Of_Item_Sold
FROM sku_data
WHERE SKU IN
(
   SELECT SKU
   FROM order_item
   WHERE OrderNumber IN
   (
      SELECT OrderNumber
      FROM retail_order
      WHERE OrderMonth = 'December'
      AND OrderYear = 2017))
GROUP BY Buyer,SKU_Description;
```



Showing rows 0 - 3 (4 total, Query took 0.0022 seconds.)

```
SELECT `Buyer`,`SKU_Description`,COUNT(SKU) AS Number_Of_Item_Sold FROM sku_data WHERE SKU IN ( SELECT SKU FROM order_item WHERE OrderNumber IN ( SELECT OrderNumber FROM retail_order WHERE OrderMonth = 'December' AND OrderYear = 2017)) GROUP BY Buyer,SKU_Description
```

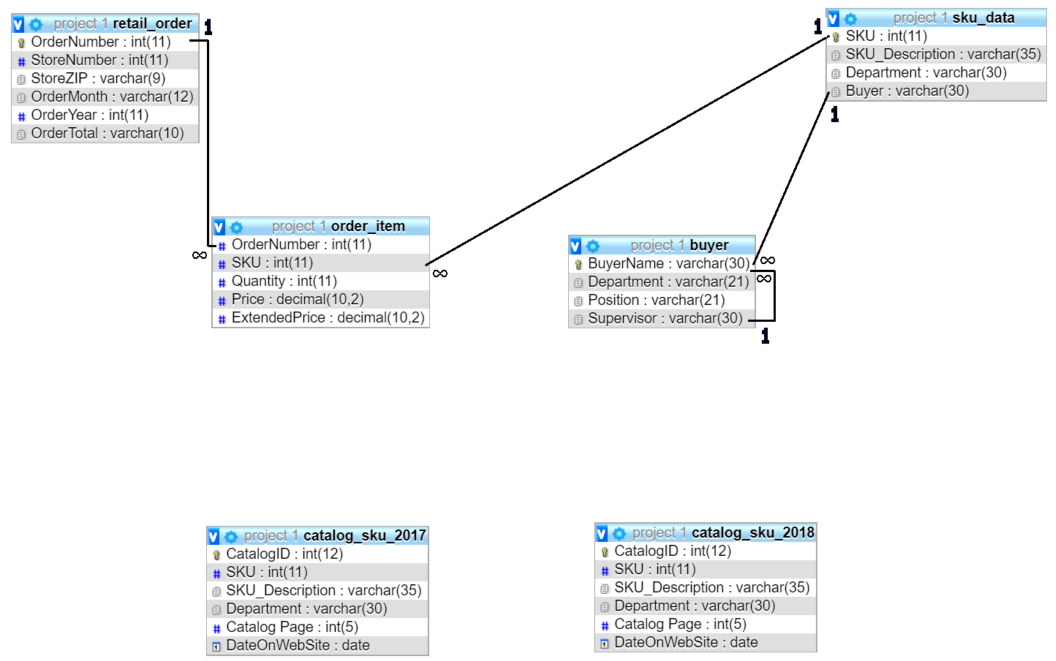
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

Buyer	SKU_Description	Number_Of_Item_Sold
Cindy Lo	Half-dome Tent	1
Cindy Lo	Half-dome Tent Vestibule	1
Nancy Meyers	Dive Mask, Med Clear	1
Nancy Meyers	Dive Mask, Small Clear	1

E/R Model



Chapter 3 – Oracle

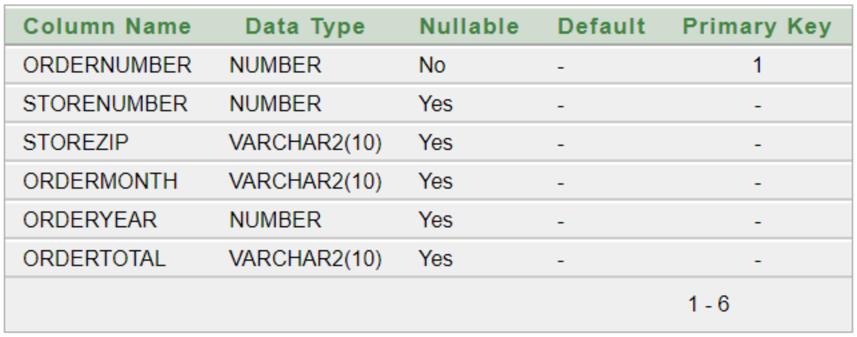
3.1. Project Description

The goal of project 3 was to practice using Oracle as a DBMS. Thus, the project was completed using Oracle and MySQL. To complete project 3 the tables created in projects 1 and 2 were recreated in Oracle. All of the same data was put into the tables and the same relations were made as well. After all of the data was entered 8 non-trivial SQL statements were written to query the tables, along with an English statement to explain each query. These statements along with the resulting output may be seen in *3.2.* below. An E/R model was also created which allows for one to visually see the relations among tables. The project was successful in allowing one to become familiar with the Oracle DBMS. I found that Oracle was much easier to use than PHP.

3.2. Tables/Input/Queries and Associated Output

<u>Part A:</u>

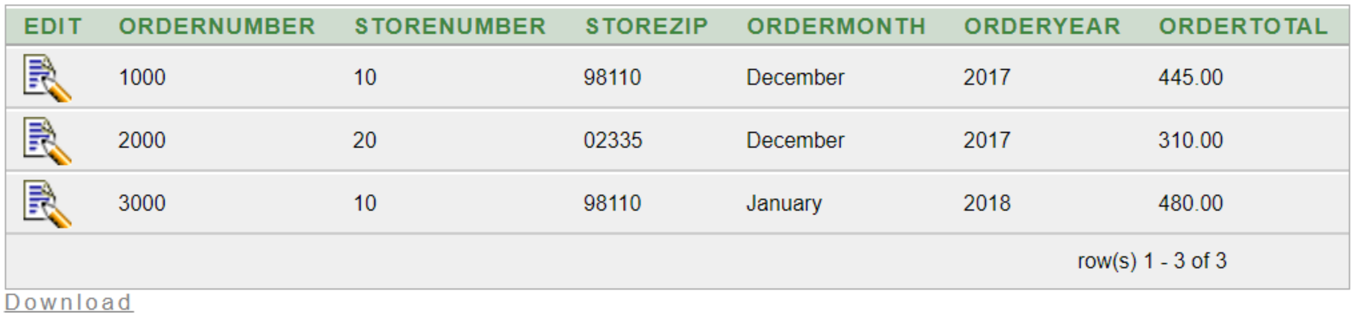
Retail_Order:



Column Name	Data Type	Nullable	Default	Primary Key
ORDERNUMBER	NUMBER	No	-	1
STORENUMBER	NUMBER	Yes	-	-
STOREZIP	VARCHAR2(10)	Yes	-	-
ORDERMONTH	VARCHAR2(10)	Yes	-	-
ORDERYEAR	NUMBER	Yes	-	-
ORDERTOTAL	VARCHAR2(10)	Yes	-	-

1 - 6

Retail_Order Table Structure



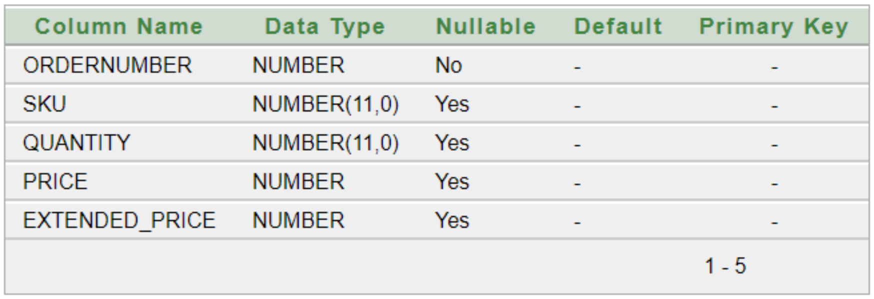
EDIT	ORDERNUMBER	STORENUMBER	STOREZIP	ORDERMONTH	ORDERYEAR	ORDERTOTAL
	1000	10	98110	December	2017	445.00
	2000	20	02335	December	2017	310.00
	3000	10	98110	January	2018	480.00

row(s) 1 - 3 of 3

Download

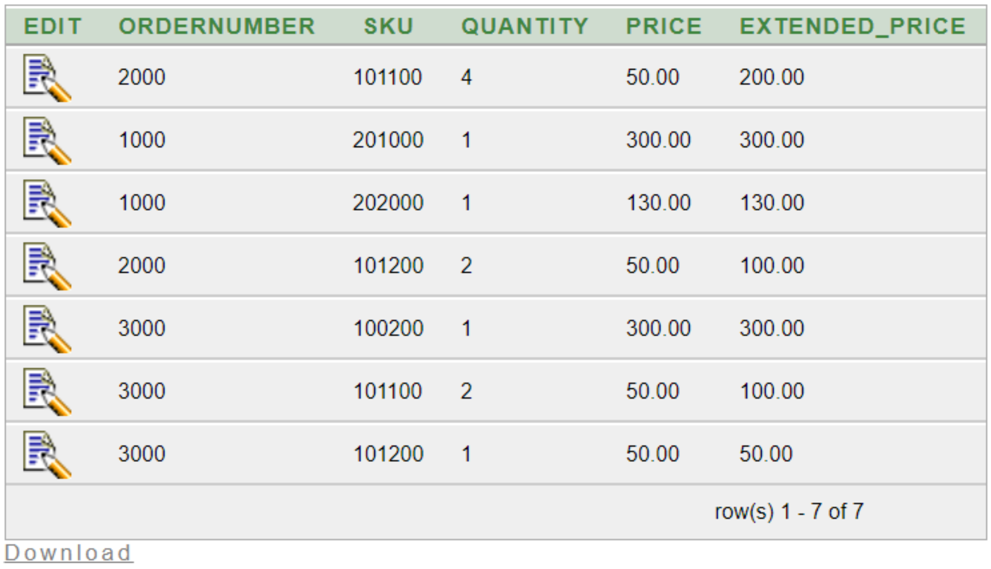
Retail_Order Table Data

Order_Item:



Column Name	Data Type	Nullable	Default	Primary Key
ORDERNUMBER	NUMBER	No	-	-
SKU	NUMBER(11,0)	Yes	-	-
QUANTITY	NUMBER(11,0)	Yes	-	-
PRICE	NUMBER	Yes	-	-
EXTENDED_PRICE	NUMBER	Yes	-	-
				1 - 5

Order_Item Table Structure

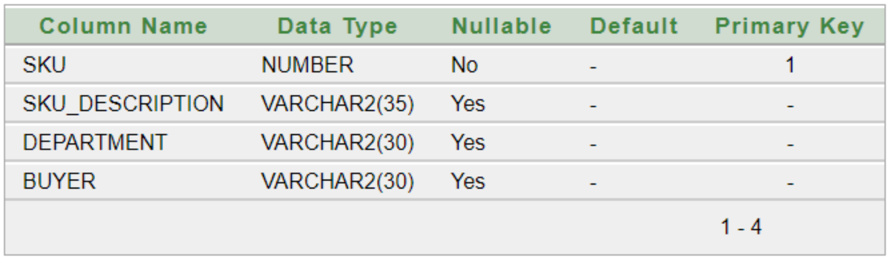


EDIT	ORDERNUMBER	SKU	QUANTITY	PRICE	EXTENDED_PRICE
✎	2000	101100	4	50.00	200.00
✎	1000	201000	1	300.00	300.00
✎	1000	202000	1	130.00	130.00
✎	2000	101200	2	50.00	100.00
✎	3000	100200	1	300.00	300.00
✎	3000	101100	2	50.00	100.00
✎	3000	101200	1	50.00	50.00
					row(s) 1 - 7 of 7

Download

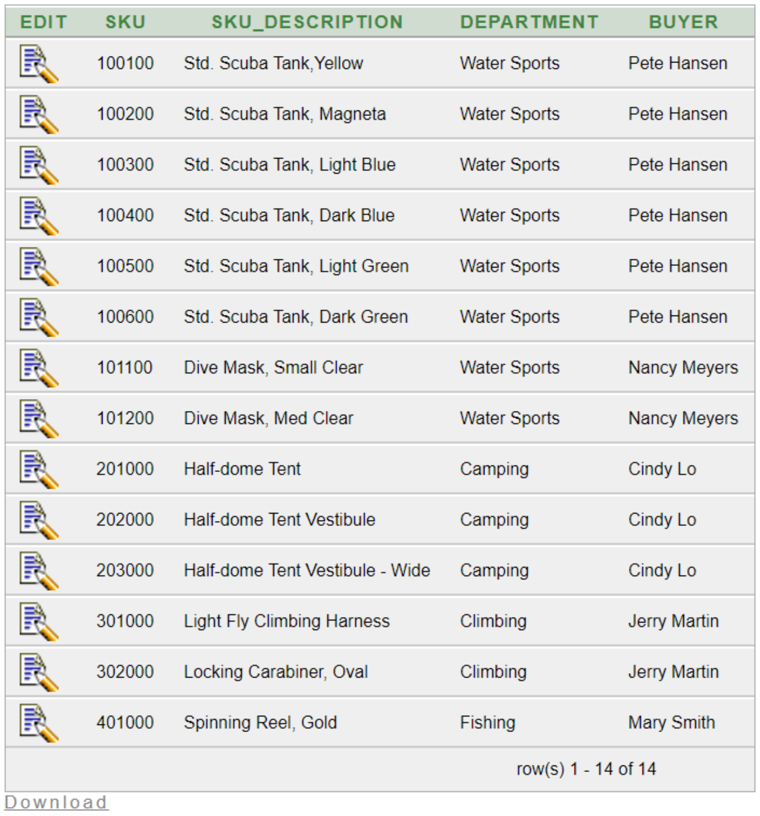
Order_Item Table Data

SKU_Data:



Column Name	Data Type	Nullable	Default	Primary Key
SKU	NUMBER	No	-	1
SKU_DESCRIPTION	VARCHAR2(35)	Yes	-	-
DEPARTMENT	VARCHAR2(30)	Yes	-	-
BUYER	VARCHAR2(30)	Yes	-	-
				1 - 4

SKU_Data Table Structure

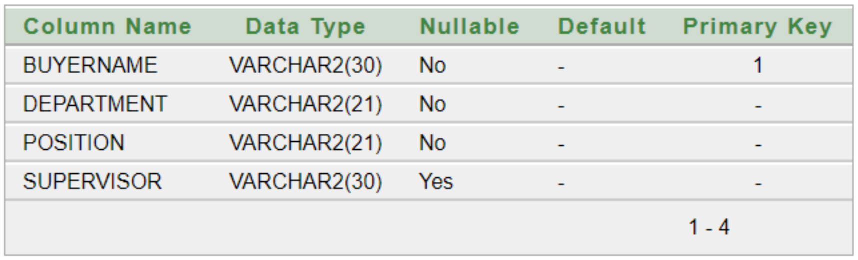


EDIT	SKU	SKU_DESCRIPTION	DEPARTMENT	BUYER
✎	100100	Std. Scuba Tank,Yellow	Water Sports	Pete Hansen
✎	100200	Std. Scuba Tank, Magneta	Water Sports	Pete Hansen
✎	100300	Std. Scuba Tank, Light Blue	Water Sports	Pete Hansen
✎	100400	Std. Scuba Tank, Dark Blue	Water Sports	Pete Hansen
✎	100500	Std. Scuba Tank, Light Green	Water Sports	Pete Hansen
✎	100600	Std. Scuba Tank, Dark Green	Water Sports	Pete Hansen
✎	101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers
✎	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers
✎	201000	Half-dome Tent	Camping	Cindy Lo
✎	202000	Half-dome Tent Vestibule	Camping	Cindy Lo
✎	203000	Half-dome Tent Vestibule - Wide	Camping	Cindy Lo
✎	301000	Light Fly Climbing Harness	Climbing	Jerry Martin
✎	302000	Locking Carabiner, Oval	Climbing	Jerry Martin
✎	401000	Spinning Reel, Gold	Fishing	Mary Smith
			row(s) 1 - 14 of 14	

Download

SKU_Data Table Data

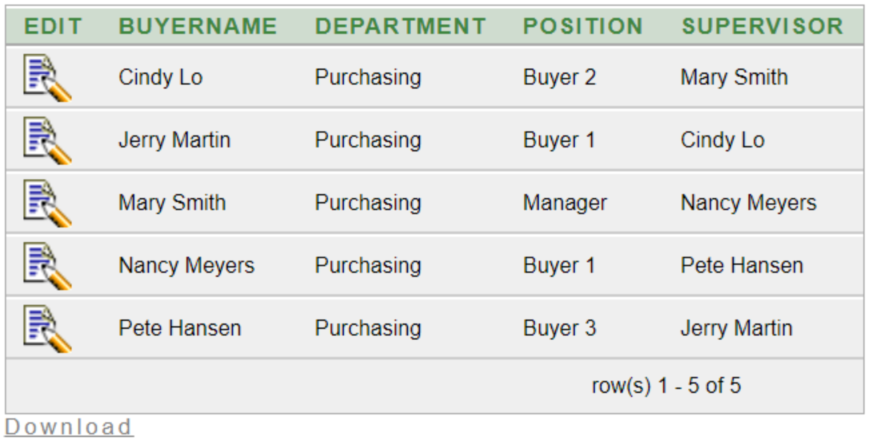
Buyer:



Column Name	Data Type	Nullable	Default	Primary Key
BUYERNAME	VARCHAR2(30)	No	-	1
DEPARTMENT	VARCHAR2(21)	No	-	-
POSITION	VARCHAR2(21)	No	-	-
SUPERVISOR	VARCHAR2(30)	Yes	-	-

1 - 4

Buyer Table Structure



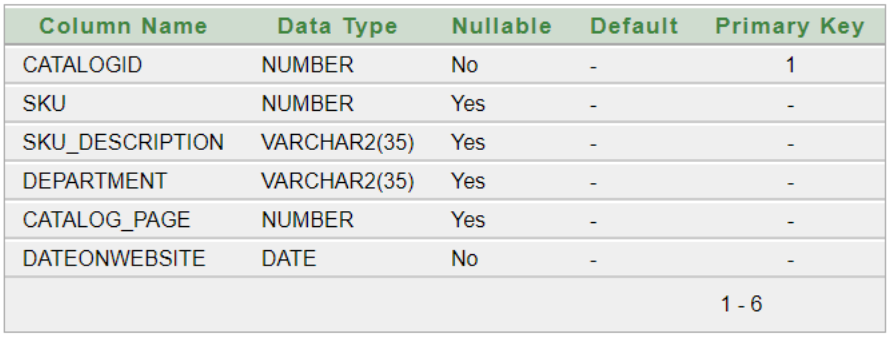
EDIT	BUYERNAME	DEPARTMENT	POSITION	SUPERVISOR
	Cindy Lo	Purchasing	Buyer 2	Mary Smith
	Jerry Martin	Purchasing	Buyer 1	Cindy Lo
	Mary Smith	Purchasing	Manager	Nancy Meyers
	Nancy Meyers	Purchasing	Buyer 1	Pete Hansen
	Pete Hansen	Purchasing	Buyer 3	Jerry Martin

row(s) 1 - 5 of 5

Download

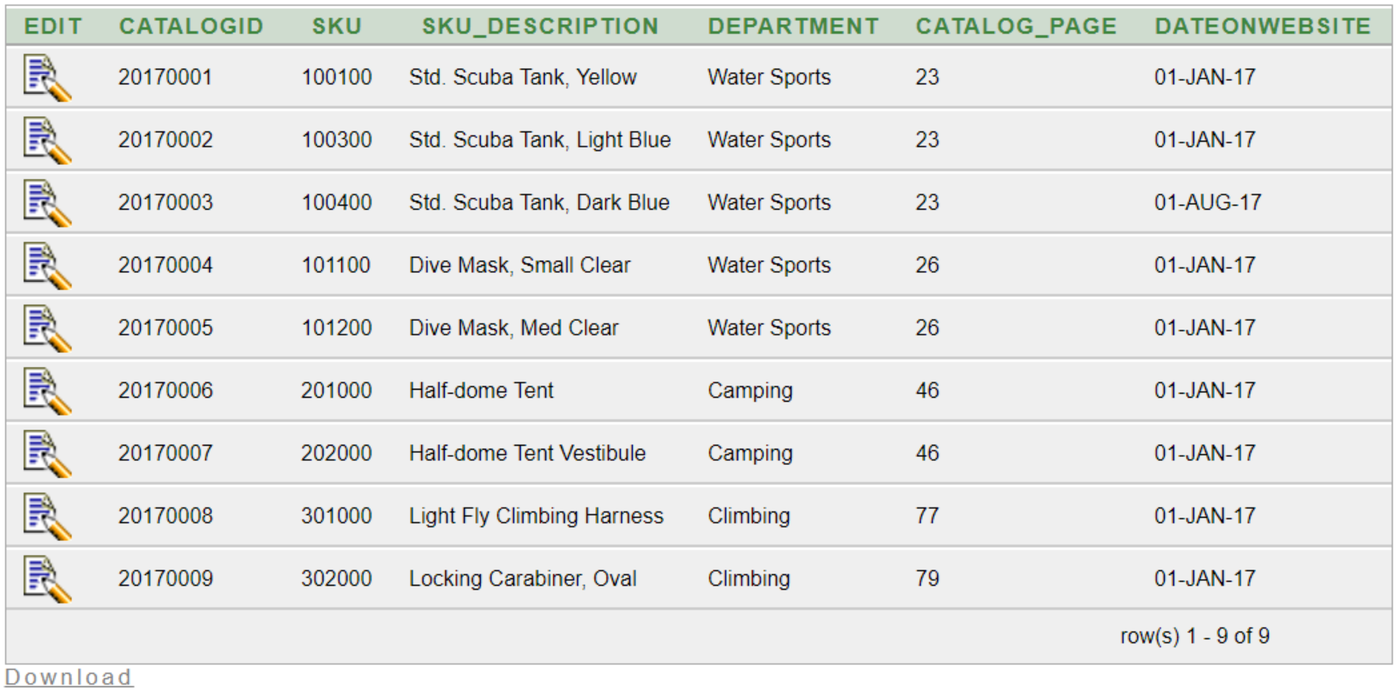
Buyer Table Data

Catalog_SKU_2017:



Column Name	Data Type	Nullable	Default	Primary Key
CATALOGID	NUMBER	No	-	1
SKU	NUMBER	Yes	-	-
SKU_DESCRIPTION	VARCHAR2(35)	Yes	-	-
DEPARTMENT	VARCHAR2(35)	Yes	-	-
CATALOG_PAGE	NUMBER	Yes	-	-
DATEONWEBSITE	DATE	No	-	-
				1 - 6

Catalog_SKU_2017 Table Structure

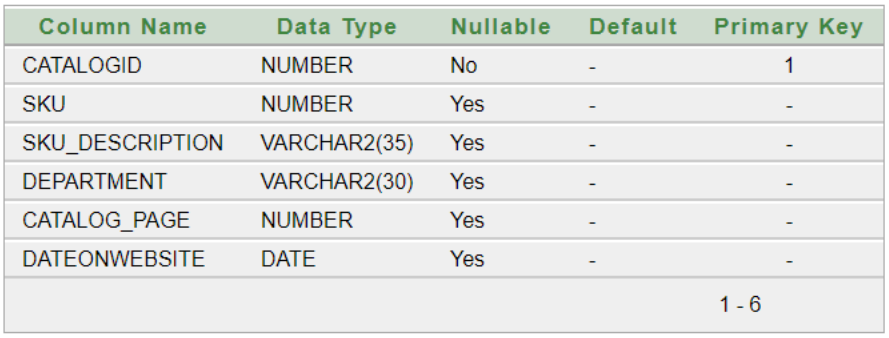


EDIT	CATALOGID	SKU	SKU_DESCRIPTION	DEPARTMENT	CATALOG_PAGE	DATEONWEBSITE
📝	20170001	100100	Std. Scuba Tank, Yellow	Water Sports	23	01-JAN-17
📝	20170002	100300	Std. Scuba Tank, Light Blue	Water Sports	23	01-JAN-17
📝	20170003	100400	Std. Scuba Tank, Dark Blue	Water Sports	23	01-AUG-17
📝	20170004	101100	Dive Mask, Small Clear	Water Sports	26	01-JAN-17
📝	20170005	101200	Dive Mask, Med Clear	Water Sports	26	01-JAN-17
📝	20170006	201000	Half-dome Tent	Camping	46	01-JAN-17
📝	20170007	202000	Half-dome Tent Vestibule	Camping	46	01-JAN-17
📝	20170008	301000	Light Fly Climbing Harness	Climbing	77	01-JAN-17
📝	20170009	302000	Locking Carabiner, Oval	Climbing	79	01-JAN-17
						row(s) 1 - 9 of 9

Download

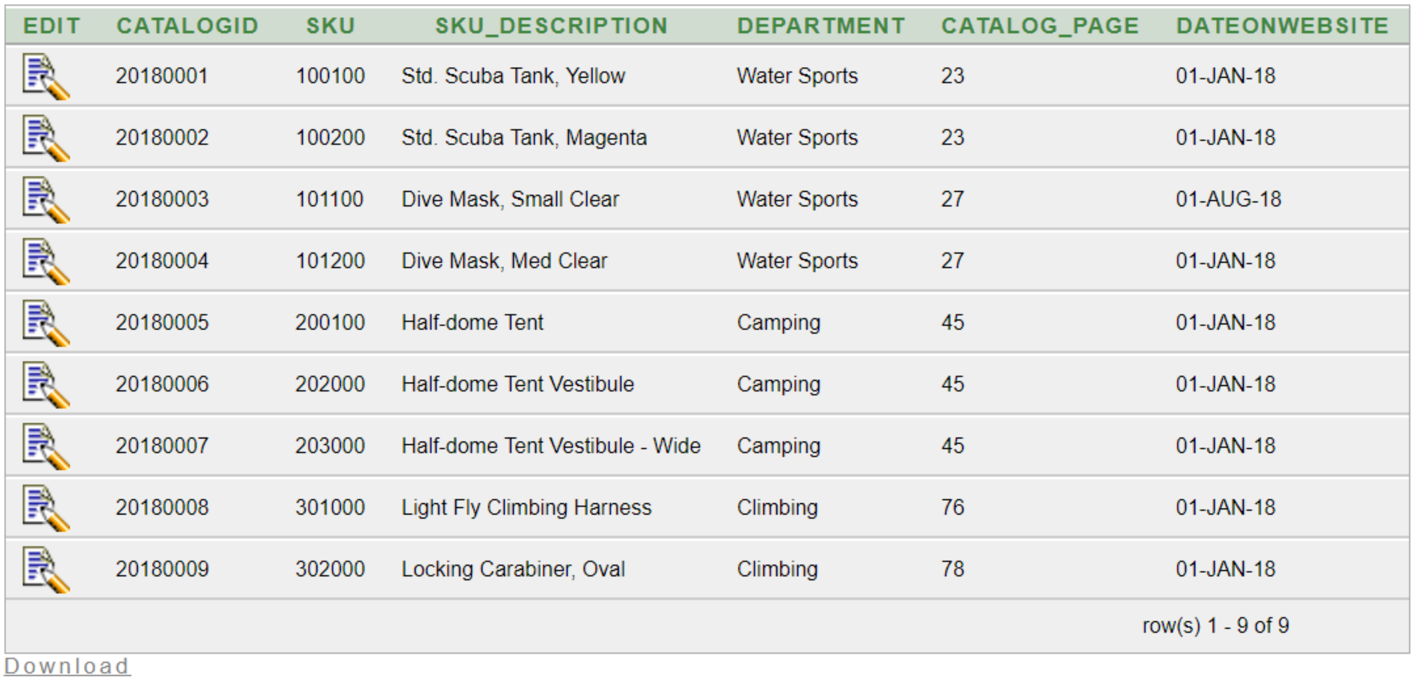
Catalog_SKU_2017 Table Data

Catalog_SKU_2018:



Column Name	Data Type	Nullable	Default	Primary Key
CATALOGID	NUMBER	No	-	1
SKU	NUMBER	Yes	-	-
SKU_DESCRIPTION	VARCHAR2(35)	Yes	-	-
DEPARTMENT	VARCHAR2(30)	Yes	-	-
CATALOG_PAGE	NUMBER	Yes	-	-
DATEONWEBSITE	DATE	Yes	-	-
				1 - 6

Catalog_SKU_2018 Table Structure



EDIT	CATALOGID	SKU	SKU_DESCRIPTION	DEPARTMENT	CATALOG_PAGE	DATEONWEBSITE
✎	20180001	100100	Std. Scuba Tank, Yellow	Water Sports	23	01-JAN-18
✎	20180002	100200	Std. Scuba Tank, Magenta	Water Sports	23	01-JAN-18
✎	20180003	101100	Dive Mask, Small Clear	Water Sports	27	01-AUG-18
✎	20180004	101200	Dive Mask, Med Clear	Water Sports	27	01-JAN-18
✎	20180005	200100	Half-dome Tent	Camping	45	01-JAN-18
✎	20180006	202000	Half-dome Tent Vestibule	Camping	45	01-JAN-18
✎	20180007	203000	Half-dome Tent Vestibule - Wide	Camping	45	01-JAN-18
✎	20180008	301000	Light Fly Climbing Harness	Climbing	76	01-JAN-18
✎	20180009	302000	Locking Carabiner, Oval	Climbing	78	01-JAN-18
						row(s) 1 - 9 of 9

Download

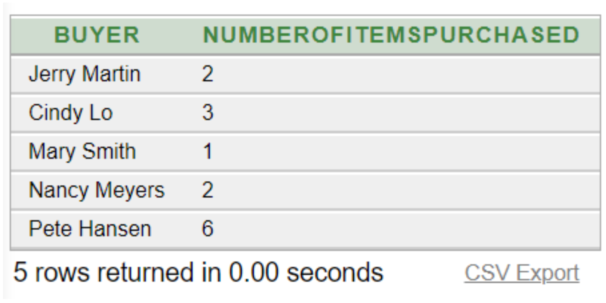
Catalog_SKU_2018 Table Data

Part B:

Query 1: Grouping

A query on the sku_data table that returns a Buyer grouped with the number of items that they have purchased in a column called NumberOfItemsPurchased.

```
SELECT Buyer, COUNT(Buyer) AS NumberOfItemsPurchased
FROM "SKU_DATA"
GROUP BY Buyer;
```



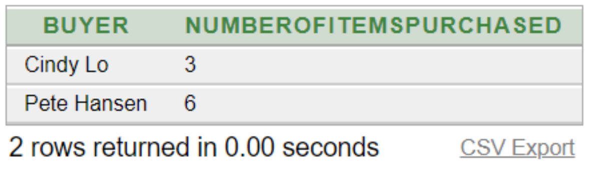
BUYER	NUMBEROFITEMSPURCHASED
Jerry Martin	2
Cindy Lo	3
Mary Smith	1
Nancy Meyers	2
Pete Hansen	6

5 rows returned in 0.00 seconds CSV Export

Query 2: Grouping

A query on the sku_data table that returns a Buyer grouped with the number of items that they have purchased in a column called NumberOfItemsPurcahased if the Buyers have purchased more than 2 items.

```
SELECT Buyer, COUNT(Buyer) AS NumberOfItemsPurchased
FROM "SKU_DATA"
GROUP BY Buyer
HAVING COUNT (Buyer)>2;
```



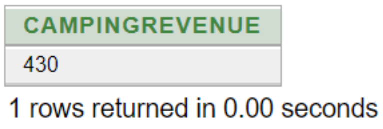
BUYER	NUMBEROFITEMSPURCHASED
Cindy Lo	3
Pete Hansen	6

2 rows returned in 0.00 seconds CSV Export

Query 3: Subquery

A query to obtain the sum of the ExtendedPrice from the order_item table for items managed by the 'Camping' Department and allocate it to a column named CampingRevenue, the sum is obtained by completing a sub-query on sku_table through SKU reference.

```
SELECT SUM(Extended_Price) AS CampingRevenue
FROM "ORDER_ITEM"
WHERE SKU IN
(
SELECT SKU
FROM "SKU_DATA"
WHERE Department = 'Camping');
```



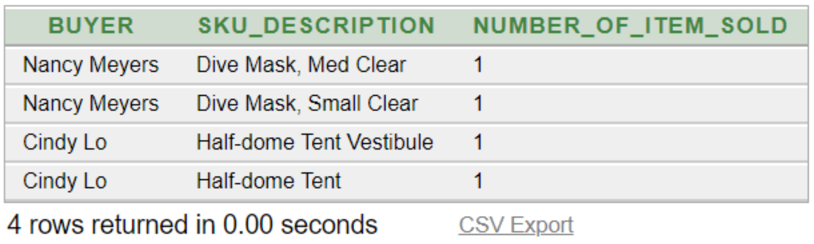
CAMPINGREVENUE
430

1 rows returned in 0.00 seconds

Query 4: Subquery

A query to obtain the number of an item sold, grouped by the items SKU_Description and Buyers name. Obtained by querying the Buyer and SKU_Description from the sku_data table and sub-querying the order information from the reail_order table using the order_item table as a reference.

```
SELECT Buyer,SKU_Description,COUNT(SKU) AS Number_Of_Item_Sold
FROM "SKU_DATA"
WHERE SKU IN
(
SELECT SKU
FROM "ORDER_ITEM"
WHERE OrderNumber IN
(
SELECT OrderNumber
FROM retail_order
WHERE OrderMonth = 'December'
AND OrderYear = 2017))
GROUP BY Buyer,SKU_Description;
```



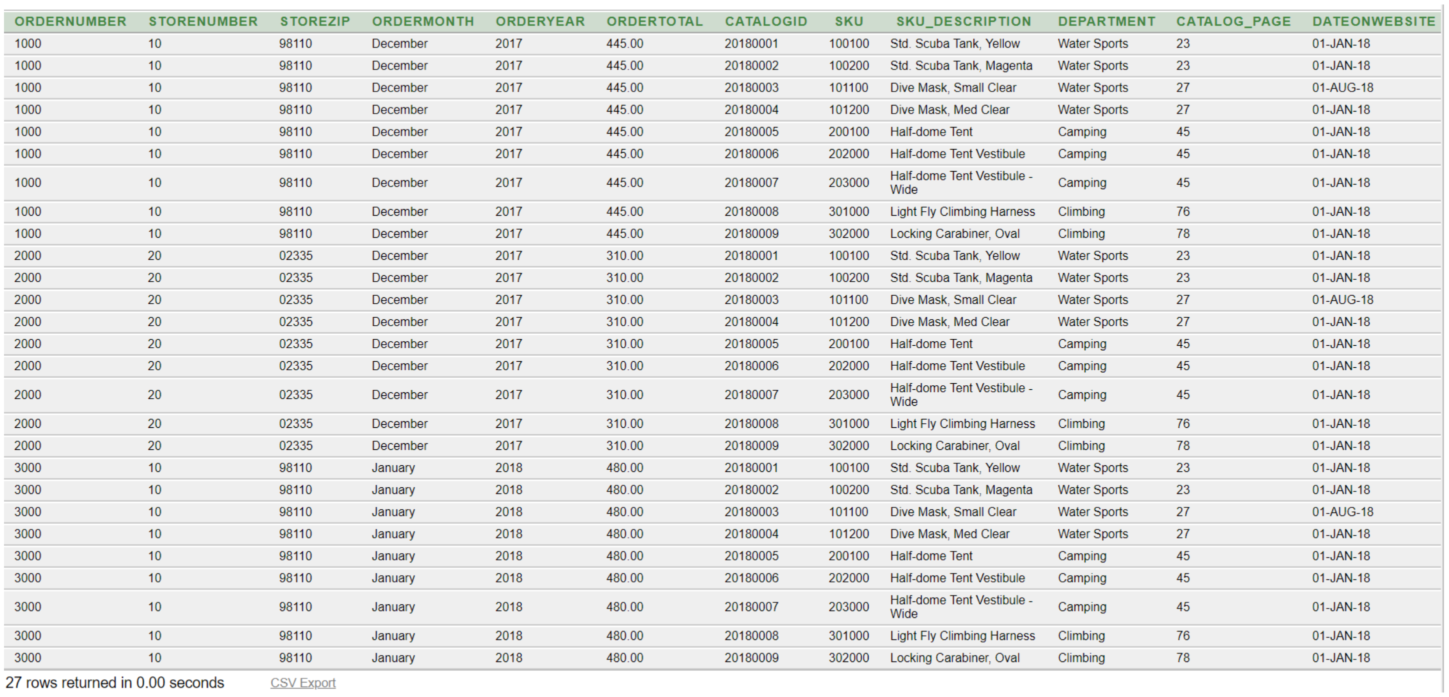
BUYER	SKU_DESCRIPTION	NUMBER_OF_ITEM_SOLD
Nancy Meyers	Dive Mask, Med Clear	1
Nancy Meyers	Dive Mask, Small Clear	1
Cindy Lo	Half-dome Tent Vestibule	1
Cindy Lo	Half-dome Tent	1

4 rows returned in 0.00 seconds CSV Export

Query 5: Cross Join

A query that selects all rows of data from the Retail_Order table (3 rows) and the Catalog_SKU_2018 table (9 rows) and cross joins them, resulting in the cartesian product of the rows in the tables, which is 27.

```
SELECT *
FROM "RETAIL_ORDER", "CATALOG_SKU_2018";
```



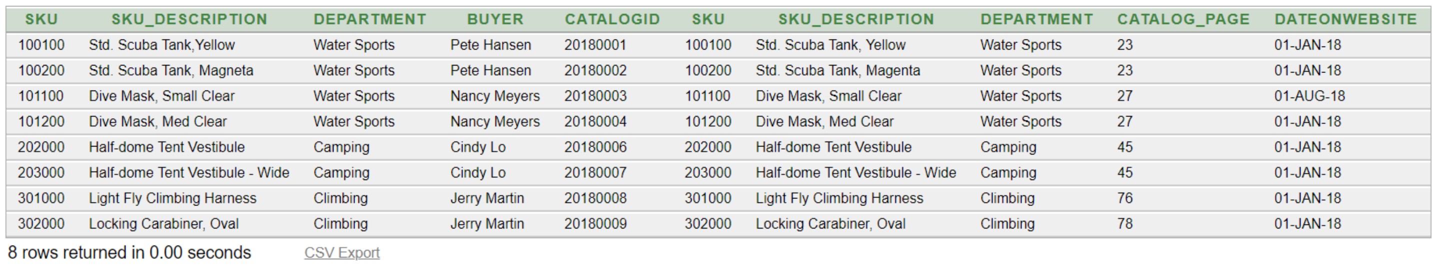
ORDERNUMBER	STORENUMBER	STOREZIP	ORDERMONTH	ORDERYEAR	ORDERTOTAL	CATALOGID	SKU	SKU_DESCRIPTION	DEPARTMENT	CATALOG_PAGE	DATEONWEBSITE
1000	10	98110	December	2017	445.00	20180001	100100	Std. Scuba Tank, Yellow	Water Sports	23	01-JAN-18
1000	10	98110	December	2017	445.00	20180002	100200	Std. Scuba Tank, Magenta	Water Sports	23	01-JAN-18
1000	10	98110	December	2017	445.00	20180003	101100	Dive Mask, Small Clear	Water Sports	27	01-AUG-18
1000	10	98110	December	2017	445.00	20180004	101200	Dive Mask, Med Clear	Water Sports	27	01-JAN-18
1000	10	98110	December	2017	445.00	20180005	200100	Half-dome Tent	Camping	45	01-JAN-18
1000	10	98110	December	2017	445.00	20180006	202000	Half-dome Tent Vestibule	Camping	45	01-JAN-18
1000	10	98110	December	2017	445.00	20180007	203000	Half-dome Tent Vestibule - Wide	Camping	45	01-JAN-18
1000	10	98110	December	2017	445.00	20180008	301000	Light Fly Climbing Harness	Climbing	76	01-JAN-18
1000	10	98110	December	2017	445.00	20180009	302000	Locking Carabiner, Oval	Climbing	78	01-JAN-18
2000	20	02335	December	2017	310.00	20180001	100100	Std. Scuba Tank, Yellow	Water Sports	23	01-JAN-18
2000	20	02335	December	2017	310.00	20180002	100200	Std. Scuba Tank, Magenta	Water Sports	23	01-JAN-18
2000	20	02335	December	2017	310.00	20180003	101100	Dive Mask, Small Clear	Water Sports	27	01-AUG-18
2000	20	02335	December	2017	310.00	20180004	101200	Dive Mask, Med Clear	Water Sports	27	01-JAN-18
2000	20	02335	December	2017	310.00	20180005	200100	Half-dome Tent	Camping	45	01-JAN-18
2000	20	02335	December	2017	310.00	20180006	202000	Half-dome Tent Vestibule	Camping	45	01-JAN-18
2000	20	02335	December	2017	310.00	20180007	203000	Half-dome Tent Vestibule - Wide	Camping	45	01-JAN-18
2000	20	02335	December	2017	310.00	20180008	301000	Light Fly Climbing Harness	Climbing	76	01-JAN-18
2000	20	02335	December	2017	310.00	20180009	302000	Locking Carabiner, Oval	Climbing	78	01-JAN-18
3000	10	98110	January	2018	480.00	20180001	100100	Std. Scuba Tank, Yellow	Water Sports	23	01-JAN-18
3000	10	98110	January	2018	480.00	20180002	100200	Std. Scuba Tank, Magenta	Water Sports	23	01-JAN-18
3000	10	98110	January	2018	480.00	20180003	101100	Dive Mask, Small Clear	Water Sports	27	01-AUG-18
3000	10	98110	January	2018	480.00	20180004	101200	Dive Mask, Med Clear	Water Sports	27	01-JAN-18
3000	10	98110	January	2018	480.00	20180005	200100	Half-dome Tent	Camping	45	01-JAN-18
3000	10	98110	January	2018	480.00	20180006	202000	Half-dome Tent Vestibule	Camping	45	01-JAN-18
3000	10	98110	January	2018	480.00	20180007	203000	Half-dome Tent Vestibule - Wide	Camping	45	01-JAN-18
3000	10	98110	January	2018	480.00	20180008	301000	Light Fly Climbing Harness	Climbing	76	01-JAN-18
3000	10	98110	January	2018	480.00	20180009	302000	Locking Carabiner, Oval	Climbing	78	01-JAN-18

27 rows returned in 0.00 seconds   CSV Export

Query 6: Implicit Join

A query that selects all rows of data from the SKU_Data table and the Catalog_SKU_2018 table and performs and inner implicit join by selecting rows where the SKU in SKU_Data matches the SKU in Catalog_SKU_2018.

```
SELECT *
FROM "SKU_DATA", "CATALOG_SKU_2018"
WHERE SKU_DATA.SKU = CATALOG_SKU_2018.SKU;
```



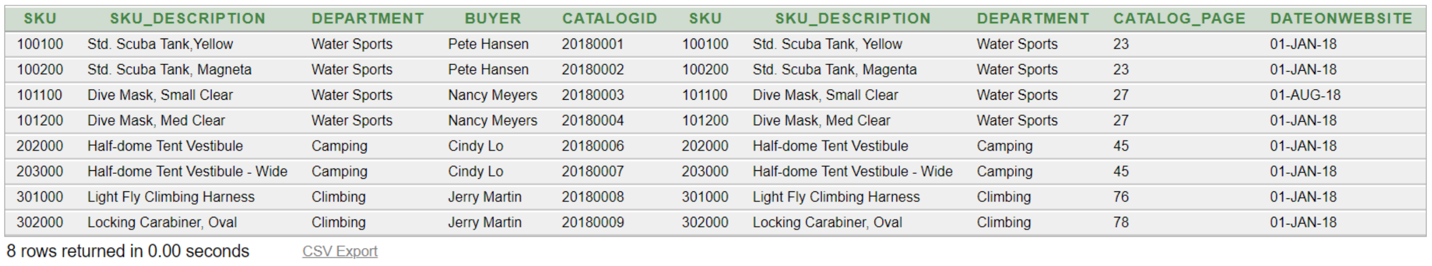
SKU	SKU_DESCRIPTION	DEPARTMENT	BUYER	CATALOGID	SKU	SKU_DESCRIPTION	DEPARTMENT	CATALOG_PAGE	DATEONWEBSITE
100100	Std. Scuba Tank,Yellow	Water Sports	Pete Hansen	20180001	100100	Std. Scuba Tank, Yellow	Water Sports	23	01-JAN-18
100200	Std. Scuba Tank, Magneta	Water Sports	Pete Hansen	20180002	100200	Std. Scuba Tank, Magenta	Water Sports	23	01-JAN-18
101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers	20180003	101100	Dive Mask, Small Clear	Water Sports	27	01-AUG-18
101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers	20180004	101200	Dive Mask, Med Clear	Water Sports	27	01-JAN-18
202000	Half-dome Tent Vestibule	Camping	Cindy Lo	20180006	202000	Half-dome Tent Vestibule	Camping	45	01-JAN-18
203000	Half-dome Tent Vestibule - Wide	Camping	Cindy Lo	20180007	203000	Half-dome Tent Vestibule - Wide	Camping	45	01-JAN-18
301000	Light Fly Climbing Harness	Climbing	Jerry Martin	20180008	301000	Light Fly Climbing Harness	Climbing	76	01-JAN-18
302000	Locking Carabiner, Oval	Climbing	Jerry Martin	20180009	302000	Locking Carabiner, Oval	Climbing	78	01-JAN-18

8 rows returned in 0.00 seconds   CSV Export

Query 7: Explicit Join

A query that selects all rows of data from the SKU_Data table and the Catalog_SKU_2018 table and performs and explicit join by selecting rows where the SKU in SKU_Data matches the SKU in Catalog_SKU_2018.

```
SELECT *
FROM "SKU_DATA" JOIN "CATALOG_SKU_2018"
ON SKU_DATA.SKU = CATALOG_SKU_2018.SKU;
```



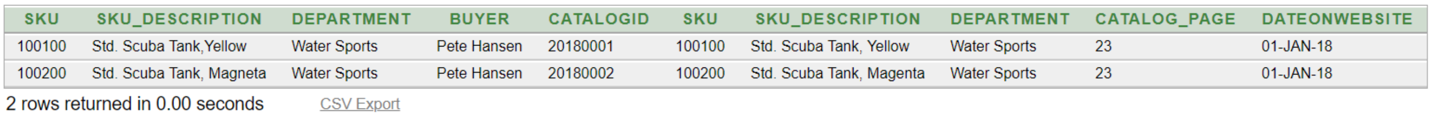
SKU	SKU_DESCRIPTION	DEPARTMENT	BUYER	CATALOGID	SKU	SKU_DESCRIPTION	DEPARTMENT	CATALOG_PAGE	DATEONWEBSITE
100100	Std. Scuba Tank,Yellow	Water Sports	Pete Hansen	20180001	100100	Std. Scuba Tank, Yellow	Water Sports	23	01-JAN-18
100200	Std. Scuba Tank, Magneta	Water Sports	Pete Hansen	20180002	100200	Std. Scuba Tank, Magenta	Water Sports	23	01-JAN-18
101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers	20180003	101100	Dive Mask, Small Clear	Water Sports	27	01-AUG-18
101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers	20180004	101200	Dive Mask, Med Clear	Water Sports	27	01-JAN-18
202000	Half-dome Tent Vestibule	Camping	Cindy Lo	20180006	202000	Half-dome Tent Vestibule	Camping	45	01-JAN-18
203000	Half-dome Tent Vestibule - Wide	Camping	Cindy Lo	20180007	203000	Half-dome Tent Vestibule - Wide	Camping	45	01-JAN-18
301000	Light Fly Climbing Harness	Climbing	Jerry Martin	20180008	301000	Light Fly Climbing Harness	Climbing	76	01-JAN-18
302000	Locking Carabiner, Oval	Climbing	Jerry Martin	20180009	302000	Locking Carabiner, Oval	Climbing	78	01-JAN-18

8 rows returned in 0.00 seconds CSV Export

Query 8: Explicit Join

A query that selects all rows of data from the SKU_Data table and the Catalog_SKU_2018 table and performs and explicit join by selecting rows where the SKU in SKU_Data matches the SKU in Catalog_SKU_2018 and where the Buyer is Pete Hansen.

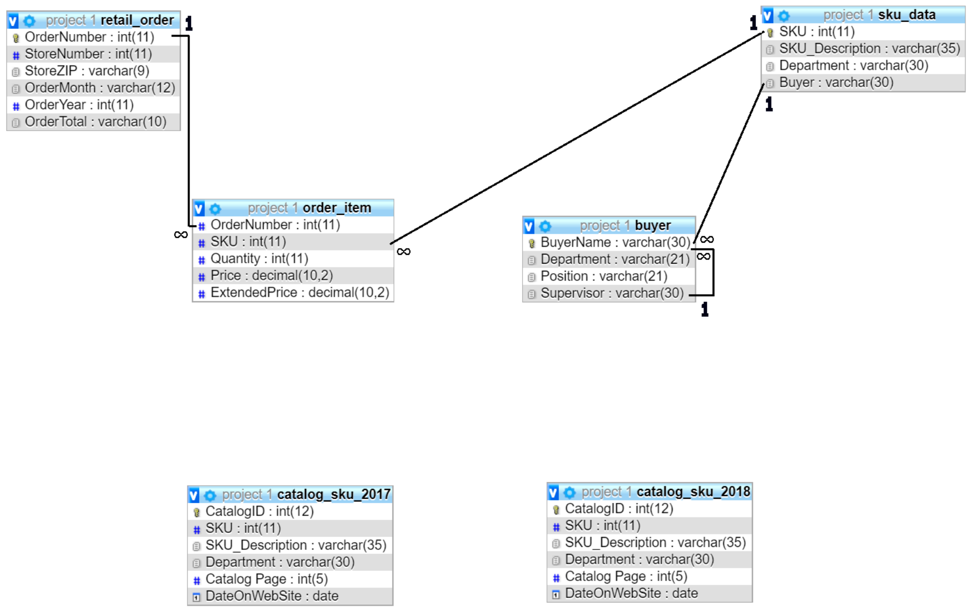
```
SELECT *
FROM "SKU_DATA" JOIN "CATALOG_SKU_2018"
ON SKU_DATA.SKU = CATALOG_SKU_2018.SKU
WHERE Buyer = 'Pete Hansen';
```



SKU	SKU_DESCRIPTION	DEPARTMENT	BUYER	CATALOGID	SKU	SKU_DESCRIPTION	DEPARTMENT	CATALOG_PAGE	DATEONWEBSITE
100100	Std. Scuba Tank,Yellow	Water Sports	Pete Hansen	20180001	100100	Std. Scuba Tank, Yellow	Water Sports	23	01-JAN-18
100200	Std. Scuba Tank, Magneta	Water Sports	Pete Hansen	20180002	100200	Std. Scuba Tank, Magenta	Water Sports	23	01-JAN-18

2 rows returned in 0.00 seconds CSV Export

E/R Model


project 1 retail_order
OrderNumber : int(11)
StoreNumber : int(11)
StoreZIP : varchar(9)
OrderMonth : varchar(12)
OrderYear : int(11)
OrderTotal : varchar(10)
project 1 sku_data
SKU : int(11)
SKU_Description : varchar(35)
Department : varchar(30)
Buyer : varchar(30)
project 1 order_item
OrderNumber : int(11)
SKU : int(11)
Quantity : int(11)
Price : decimal(10,2)
ExtendedPrice : decimal(10,2)
project 1 buyer
BuyerName : varchar(30)
Department : varchar(21)
Position : varchar(21)
Supervisor : varchar(30)
project 1 catalog_sku_2017
CatalogID : int(12)
SKU : int(11)
SKU_Description : varchar(35)
Department : varchar(30)
Catalog Page : int(5)
DateOnWebSite : date
project 1 catalog_sku_2018
CatalogID : int(12)
SKU : int(11)
SKU_Description : varchar(35)
Department : varchar(30)
Catalog Page : int(5)
DateOnWebSite : date
1
∞
1
∞
1
∞
∞
1

Chapter 4 – Create and Insert for Morgan Importing

4.1. Project Description

The goal of project 4 was to practice coding in SQL to create tables and insert data into tables. The project was completed using Oracle and MySQL. To complete this project, three tables needed to be created and have data inserted into them. The code to create the item table, shipment table, and shipment item table for the Morgan Importing Database, may be seen in *4.2.*, along with the code written to insert data into the tables. After all of the data was entered 8 non-trivial SQL statements were written to query the tables, along with an English statement to explain each query. These statements along with the resulting output may be seen in *4.2.* below. An E/R model was also created which allows for one to visually see the relations among tables. The project was successful in teaching one how to write create and insert statements in SQL to create tables and insert data into the created tables.

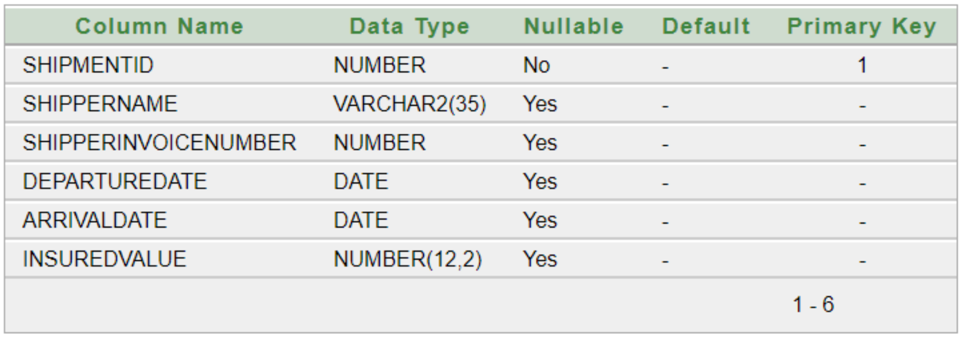
4.2. Tables/Input/Queries and Associated Output

Part A:

Item:

Create Statement:

```
CREATE TABLE  "ITEM"
   (     "ITEMID" NUMBER,
         "DESCRIPTION" VARCHAR2(255),
         "PURCHASEDATE" DATE,
         "STORE" VARCHAR2(50),
         "CITY" VARCHAR2(35),
         "QUANTITY" NUMBER,
         "LOCALCURRENCYAMOUNT" NUMBER(18,2),
         "EXCHANGERATE" NUMBER(12,6),
          CONSTRAINT "ITEM_PK" PRIMARY KEY ("ITEMID") ENABLE
   )
```

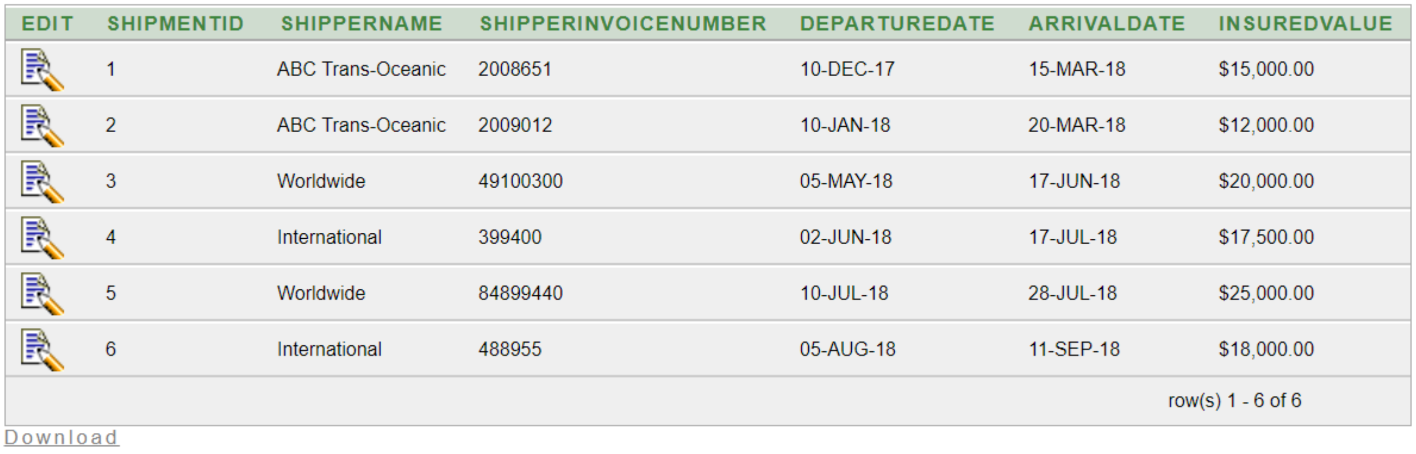


Column Name	Data Type	Nullable	Default	Primary Key
SHIPMENTID	NUMBER	No	-	1
SHIPPERNAME	VARCHAR2(35)	Yes	-	-
SHIPPERINVOICENUMBER	NUMBER	Yes	-	-
DEPARTUREDATE	DATE	Yes	-	-
ARRIVALDATE	DATE	Yes	-	-
INSUREDVALUE	NUMBER(12,2)	Yes	-	-
				1 - 6

Item Table Structure

Insert Statements:

```
INSERT INTO ITEM (ItemID, Description, PurchaseDate, Store, City, Quantity, LocalCurrencyAmount,
ExchangeRate)
VALUES (1, 'QE Dining Set', 07-Apr-18, 'Eastern Treasures', 'Manila', 2, 403405, 0.01774);

INSERT INTO ITEM (ItemID, Description, PurchaseDate, Store, City, Quantity, LocalCurrencyAmount,
ExchangeRate)
VALUES (2, 'Willow Serving Dishes', 15-Jul-18, 'Jade Antiques', 'Singapore', 75, 102, 0.5903);

INSERT INTO ITEM (ItemID, Description, PurchaseDate, Store, City, Quantity, LocalCurrencyAmount,
ExchangeRate)
VALUES (3, 'Large Bureau', 17-Jul-18, 'Eastern Sales', 'Singapore', 8, 2000, 0.5903);

INSERT INTO ITEM (ItemID, Description, PurchaseDate, Store, City, Quantity, LocalCurrencyAmount,
ExchangeRate)
VALUES (4, 'Brass Lamps', 20-Jul-18, 'Jade Antiques', 'Singapore', 40, 50, 0.5903);
```



EDIT	SHIPMENTID	SHIPPERNAME	SHIPPERINVOICENUMBER	DEPARTUREDATE	ARRIVALDATE	INSUREDVALUE
✎	1	ABC Trans-Oceanic	2008651	10-DEC-17	15-MAR-18	$15,000.00
✎	2	ABC Trans-Oceanic	2009012	10-JAN-18	20-MAR-18	$12,000.00
✎	3	Worldwide	49100300	05-MAY-18	17-JUN-18	$20,000.00
✎	4	International	399400	02-JUN-18	17-JUL-18	$17,500.00
✎	5	Worldwide	84899440	10-JUL-18	28-JUL-18	$25,000.00
✎	6	International	488955	05-AUG-18	11-SEP-18	$18,000.00
						row(s) 1 - 6 of 6

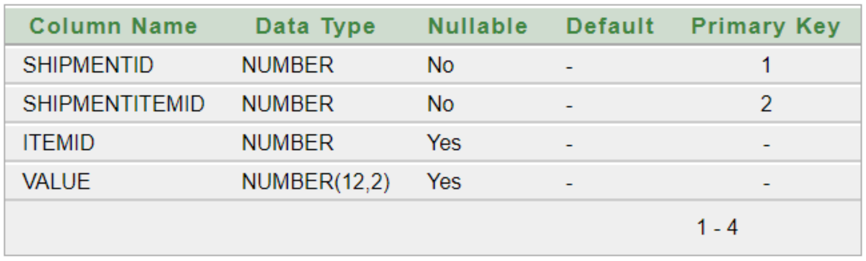
Download

Item Table Data

Shipment:

Create Statement:

```
CREATE TABLE  "SHIPMENT"
   (        "SHIPMENTID" NUMBER,
            "SHIPPERNAME" VARCHAR2(35),
            "SHIPPERINVOICENUMBER" NUMBER,
            "DEPARTUREDATE" DATE,
            "ARRIVALDATE" DATE,
            "INSUREDVALUE" NUMBER(12,2),
             CONSTRAINT "SHIPMENT_PK" PRIMARY KEY ("SHIPMENTID") ENABLE
   )
```



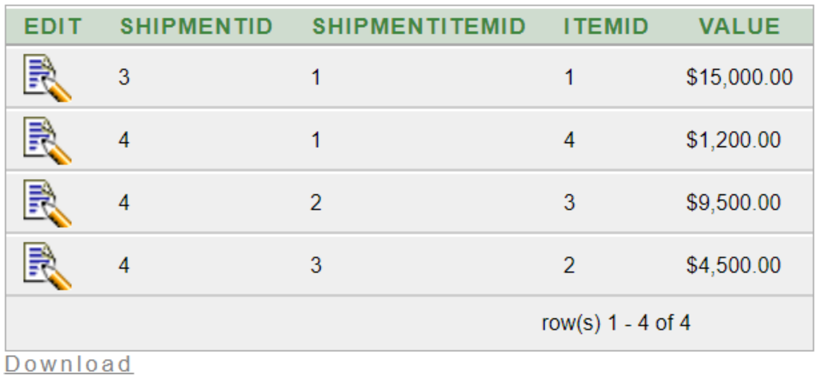
Column Name	Data Type	Nullable	Default	Primary Key
SHIPMENTID	NUMBER	No	-	1
SHIPMENTITEMID	NUMBER	No	-	2
ITEMID	NUMBER	Yes	-	-
VALUE	NUMBER(12,2)	Yes	-	-
				1 - 4

Shipment Table Structure

Insert Statements:

```
INSERT INTO SHIPMENT (ShipmentID, ShipperName, ShipperInvoiceNumber, DepartureDate, ArrivalDate,
InsuredValue)
VALUES (1, 'ABC Trans-Oceanic', 2008651, '10-Dec-17', '15-Mar-18', '$15,000.00');

INSERT INTO SHIPMENT (ShipmentID, ShipperName, ShipperInvoiceNumber, DepartureDate, ArrivalDate,
InsuredValue)
VALUES (2, 'ABC Trans-Oceanic', 2009012, '10-Jan-18', '20-Mar-18', '$12,000.00');

INSERT INTO SHIPMENT (ShipmentID, ShipperName, ShipperInvoiceNumber, DepartureDate, ArrivalDate,
InsuredValue)
```

```
VALUES (3, 'Worldwide', 49100300, '05-May-18', '17-Jun-18', '$20,000.00');

INSERT INTO SHIPMENT (ShipmentID, ShipperName, ShipperInvoiceNumber, DepartureDate, ArrivalDate,
InsuredValue)
VALUES (4, 'International', 399400, '02-Jun-18', '17-Jul-18', '$17,500.00');

INSERT INTO SHIPMENT (ShipmentID, ShipperName, ShipperInvoiceNumber, DepartureDate, ArrivalDate,
InsuredValue)
VALUES (5, 'Worldwide', 84899440, '10-Jul-18', '28-Jul-18', '$25,000.00');

INSERT INTO SHIPMENT (ShipmentID, ShipperName, ShipperInvoiceNumber, DepartureDate, ArrivalDate,
InsuredValue)
VALUES (6, 'International', 488955, '05-Aug-18', '11-Sep-18', '$18,000.00');
```



EDIT	SHIPMENTID	SHIPMENTITEMID	ITEMID	VALUE
	3	1	1	$15,000.00
	4	1	4	$1,200.00
	4	2	3	$9,500.00
	4	3	2	$4,500.00

row(s) 1 - 4 of 4

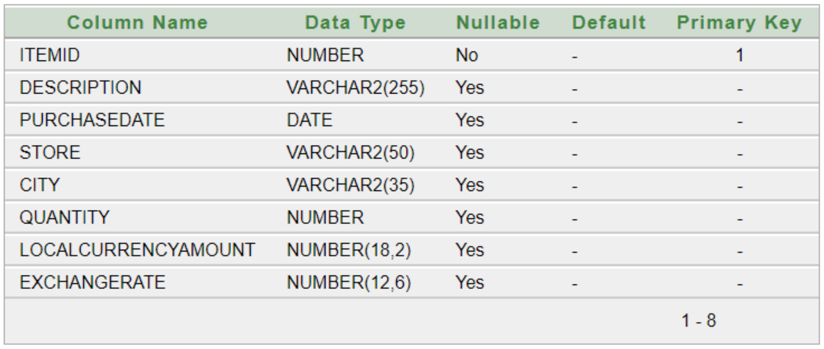
Download

Shipment Table Data

Shipment_Item:

Create Statement:

```
CREATE TABLE  "SHIPMENT_ITEM"
   (	"SHIPMENTID" NUMBER,
	"SHIPMENTITEMID" NUMBER,
	"ITEMID" NUMBER,
	"VALUE" NUMBER(12,2),
	 CONSTRAINT "SHIPMENTID_PK" PRIMARY KEY ("SHIPMENTID", "SHIPMENTITEMID")
ENABLE,
	 CONSTRAINT "FK_SHIPMENT" FOREIGN KEY ("SHIPMENTID")
	  REFERENCES  "SHIPMENT" ("SHIPMENTID") ENABLE,
	 CONSTRAINT "FK_ITEM" FOREIGN KEY ("ITEMID")
	  REFERENCES  "ITEM" ("ITEMID") ENABLE
   )
```



Column Name	Data Type	Nullable	Default	Primary Key
ITEMID	NUMBER	No	-	1
DESCRIPTION	VARCHAR2(255)	Yes	-	-
PURCHASEDATE	DATE	Yes	-	-
STORE	VARCHAR2(50)	Yes	-	-
CITY	VARCHAR2(35)	Yes	-	-
QUANTITY	NUMBER	Yes	-	-
LOCALCURRENCYAMOUNT	NUMBER(18,2)	Yes	-	-
EXCHANGERATE	NUMBER(12,6)	Yes	-	-

1 - 8

Shipment Table Structure

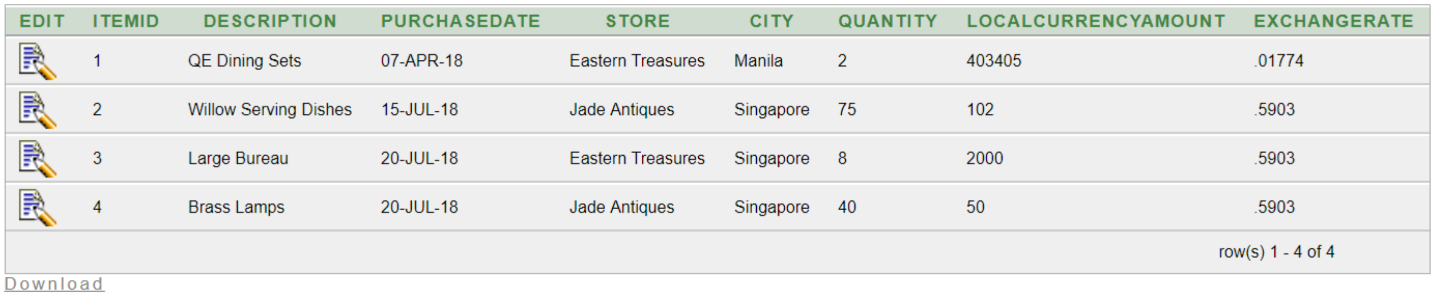
Insert Statements:

INSERT INTO SHIPMENT_ITEM (ShipmentID, ShipmentItemID, ItemID, Value)
VALUES (3, 1, 1, '$15,000.00');

INSERT INTO SHIPMENT_ITEM (ShipmentID, ShipmentItemID, ItemID, Value)
VALUES (4, 1, 4, '$1,200.00');

INSERT INTO SHIPMENT_ITEM (ShipmentID, ShipmentItemID, ItemID, Value)
VALUES (4, 2, 3, '$9,500.00');

INSERT INTO SHIPMENT_ITEM (ShipmentID, ShipmentItemID, ItemID, Value)
VALUES (4, 3, 2, '$4,500.00');



EDIT	ITEMID	DESCRIPTION	PURCHASEDATE	STORE	CITY	QUANTITY	LOCALCURRENCYAMOUNT	EXCHANGERATE
✎	1	QE Dining Sets	07-APR-18	Eastern Treasures	Manila	2	403405	.01774
✎	2	Willow Serving Dishes	15-JUL-18	Jade Antiques	Singapore	75	102	.5903
✎	3	Large Bureau	20-JUL-18	Eastern Treasures	Singapore	8	2000	.5903
✎	4	Brass Lamps	20-JUL-18	Jade Antiques	Singapore	40	50	.5903

row(s) 1 - 4 of 4

Download

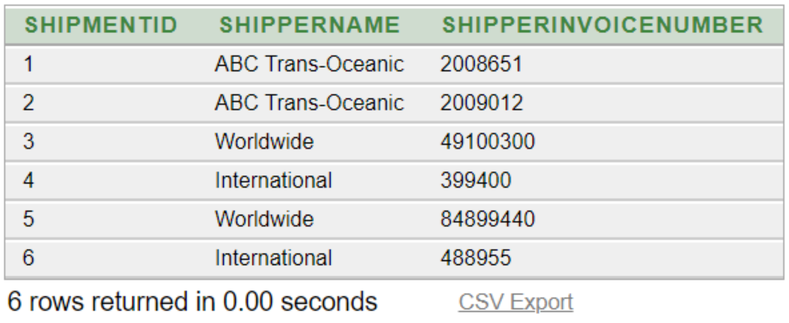
Shipment Table Data

Part B:

Query 1:
List the ShipmentID, ShipperName, and ShipperInvoiceNumber for all shipments that have an insured value greater than $10,000.00

A query to select the ShipmentID, ShipperName, and ShipperInvoiceNumber from a Shipment table entry where the InsuredValue for the entry is greater than $10,000.00.

```
SELECT ShipmentID, ShipperName, ShipperInvoiceNumber
FROM SHIPMENT
WHERE InsuredValue > 10000;
```



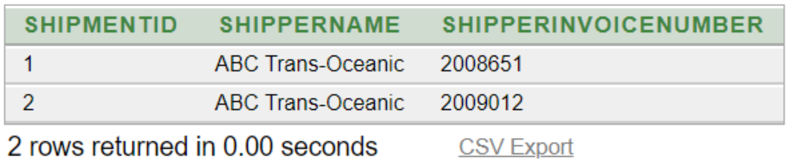
SHIPMENTID	SHIPPERNAME	SHIPPERINVOICENUMBER
1	ABC Trans-Oceanic	2008651
2	ABC Trans-Oceanic	2009012
3	Worldwide	49100300
4	International	399400
5	Worldwide	84899440
6	International	488955

6 rows returned in 0.00 seconds CSV Export

Query 2:
List the ShipmentID, ShipperName, and ShipperInvoiceNumber of all shippers whose name starts with 'AB'

A query to select the ShipmentID, ShipperName, and ShipperInvoiceNumber from Shipment table entries where the ShipperName begins with the letters 'AB'.

```
SELECT ShipmentID, ShipperName, ShipperInvoiceNumber
FROM SHIPMENT
WHERE ShipperName Like 'AB%';
```



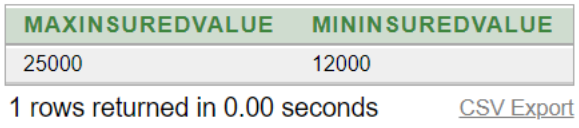
SHIPMENTID	SHIPPERNAME	SHIPPERINVOICENUMBER
1	ABC Trans-Oceanic	2008651
2	ABC Trans-Oceanic	2009012

2 rows returned in 0.00 seconds CSV Export

Query 3:
Determine the maximum and minimum InsuredValue

A query to select the maximum and minimum insured values from the InsuredValue column from the Shipement table and display them in a table as MaxInsuredValue and MinInsuredValue

SELECT MAX (InsuredValue) AS MaxInsuredValue, MIN (InsuredValue) AS MinInsuredValue
FROM SHIPMENT;



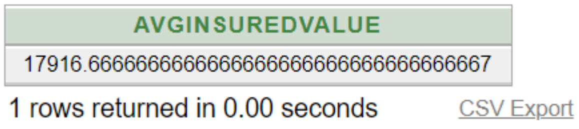
MAXINSUREDVALUE	MININSUREDVALUE
25000	12000

1 rows returned in 0.00 seconds CSV Export

Query 4:
Determine the average InsuredValue

A query to select all values from the InsuredValue column of the Shipment table and average them using the AVG command, the average is then displayed in a table as AvgInsureValue

SELECT AVG (InsuredValue) AS AvgInsuredValue
FROM SHIPMENT;



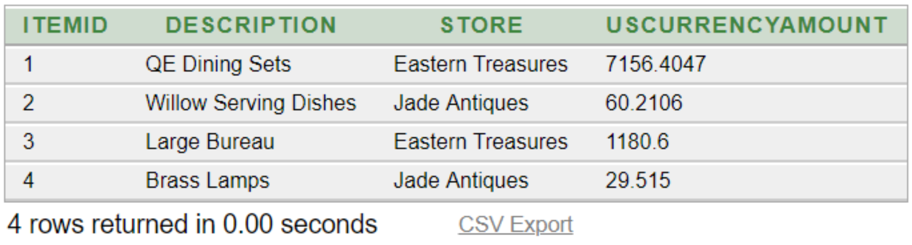
AVGINSUREDVALUE
17916.6666666666666666666666666666666667

1 rows returned in 0.00 seconds CSV Export

Query 5:
Show ItemID, Description, Store, and a calculated column named USCurrencyAmount that is equal to LocalCurrencyAmount multiplied by the ExchangeRate for all rows of ITEM

A query to select the ItemID, Description, Store, and the product of LocalCurrencyAmount and ExchangeRate from the Item table and display them in a table where the product of LocalCurrencyAmount and ExchangeRate is a column named USCurrencyAmount

SELECT ItemID, Description, Store, LocalCurrencyAmount * ExchangeRate AS USCurrencyAmount
FROM ITEM;



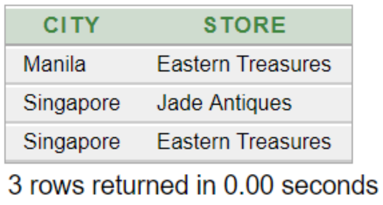
ITEMID	DESCRIPTION	STORE	USCURRENCYAMOUNT
1	QE Dining Sets	Eastern Treasures	7156.4047
2	Willow Serving Dishes	Jade Antiques	60.2106
3	Large Bureau	Eastern Treasures	1180.6
4	Brass Lamps	Jade Antiques	29.515

4 rows returned in 0.00 seconds CSV Export

Query 6:
Group item purchases by City and Store

A query to select the City and Store columns from the Item table that returns them grouped them by the unique city and store at which an item was purchased

```
SELECT City, Store
FROM ITEM
GROUP BY City, Store;
```



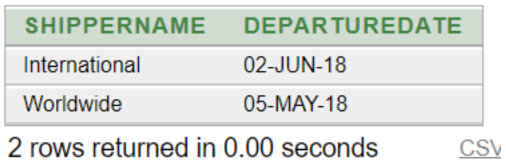
CITY	STORE
Manila	Eastern Treasures
Singapore	Jade Antiques
Singapore	Eastern Treasures

3 rows returned in 0.00 seconds

Query 7:
Show the ShipperName, ShipmentID and DepartureDate of all shipments that have an item with a value of $1,000.00 or more. Use a subquery. Present results sorted by ShipperName in ascending order and then DepartureDate in descending order

A query to obtain the ShipperName and DepartureDate from the Shipment table of all shipments that have a Value >= $1,000.00. Obtained by first querying the Shipment table and then subquerying the Shipment_Item table using the ShipmentID to retrieve quantities from Value. Results are ordered as stated in the query.

```
SELECT ShipperName, DepartureDate
FROM SHIPMENT
WHERE ShipmentID IN
(SELECT ShipmentID
FROM SHIPMENT_ITEM
WHERE Value = 1000 OR Value > 1000)
ORDER BY ShipperName, DepartureDate DESC;
```



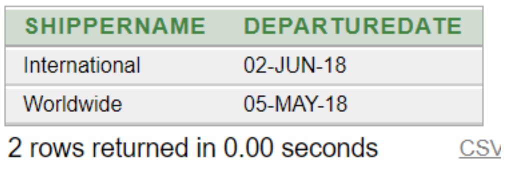
SHIPPERNAME	DEPARTUREDATE
International	02-JUN-18
Worldwide	05-MAY-18

2 rows returned in 0.00 seconds CSV

Query 8:
Show the ShipperName, ShipmentID, and DepartureDate of all shipments that have an item with a value of $1,000.00 or more. Use a join. Present results sorted by ShipperName in ascending order and then DepartureDate in descending order

A query that selects distinct ShipperNames and DepartureDates from the Shipment table and performs an explicit join by selecting the equivalent ShipmentID from the Shipment_Item table and Shipment table. The result is a table with all shipments that have a Value >= $1,000.00, ordered as stated in the query.

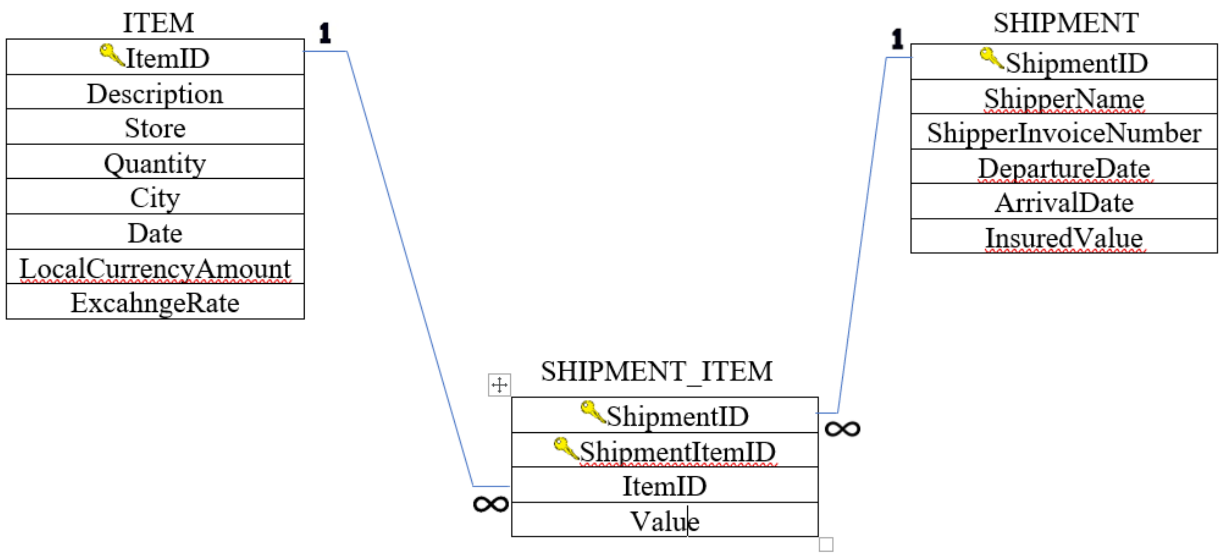
```
SELECT DISTINCT ShipperName, DepartureDate
FROM SHIPMENT, SHIPMENT_ITEM
WHERE SHIPMENT.ShipmentID = SHIPMENT_ITEM.ShipmentID
AND Value >= 1000
ORDER BY ShipperName, DepartureDate DESC;
```



SHIPPERNAME	DEPARTUREDATE
International	02-JUN-18
Worldwide	05-MAY-18

2 rows returned in 0.00 seconds CSV

E/R Model:



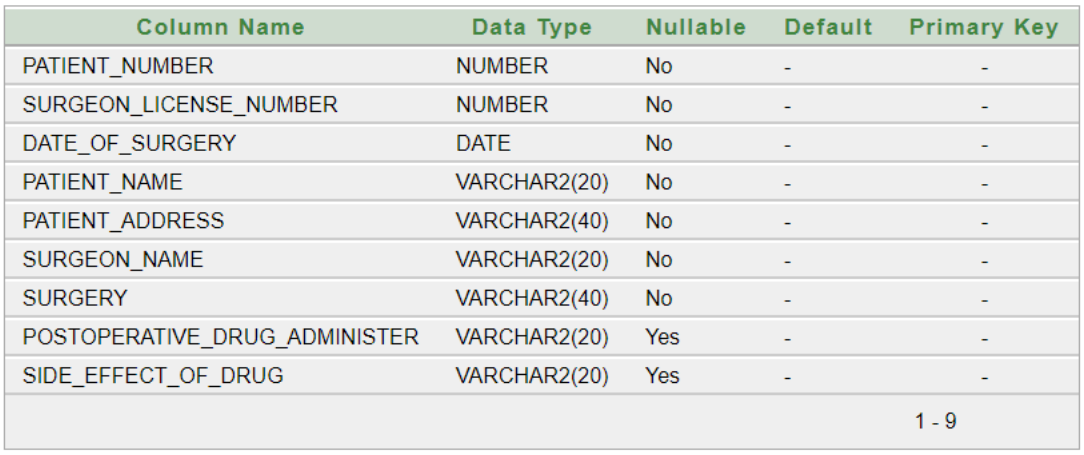
Chapter 5 – Hospital Data Table Functional Dependencies

5.1. Project Description

The goal of project 5 was to find and list all the functional dependencies in the provided simplified hospital data table. The project was completed using Oracle and MySQL. To complete the project the simplified hospital data table was created using Oracle. The functional dependencies were then derived from observing the table, and a table representing each functional dependency was created. This project was successful in teaching one how to recognize functional dependencies in any given table.

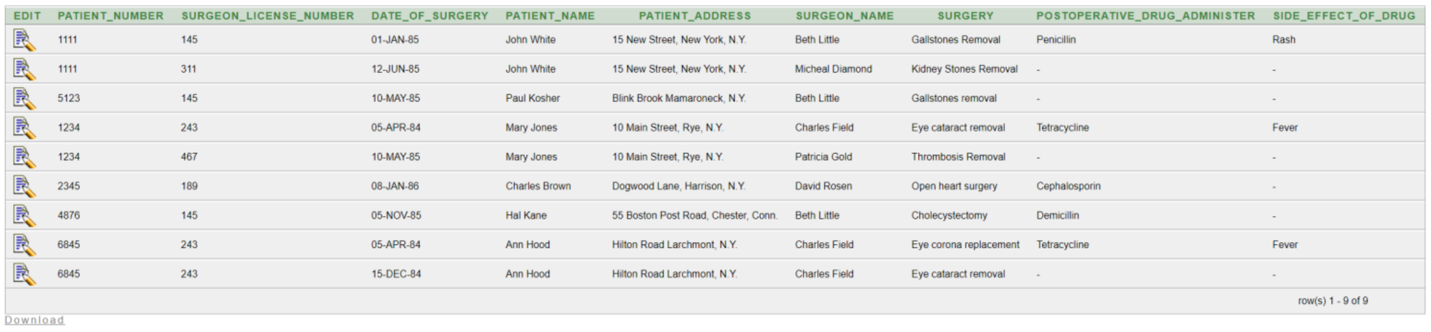
5.2. Tables/Input/Queries and Associated Output

(1) Completed Hospital Data Table



Column Name	Data Type	Nullable	Default	Primary Key
PATIENT_NUMBER	NUMBER	No	-	-
SURGEON_LICENSE_NUMBER	NUMBER	No	-	-
DATE_OF_SURGERY	DATE	No	-	-
PATIENT_NAME	VARCHAR2(20)	No	-	-
PATIENT_ADDRESS	VARCHAR2(40)	No	-	-
SURGEON_NAME	VARCHAR2(20)	No	-	-
SURGERY	VARCHAR2(40)	No	-	-
POSTOPERATIVE_DRUG_ADMINISTER	VARCHAR2(20)	Yes	-	-
SIDE_EFFECT_OF_DRUG	VARCHAR2(20)	Yes	-	-
				1 - 9

Simplified Hospital Data Structure



EDIT	PATIENT_NUMBER	SURGEON_LICENSE_NUMBER	DATE_OF_SURGERY	PATIENT_NAME	PATIENT_ADDRESS	SURGEON_NAME	SURGERY	POSTOPERATIVE_DRUG_ADMINISTER	SIDE_EFFECT_OF_DRUG
✎	1111	145	01-JAN-85	John White	15 New Street, New York, N.Y.	Beth Little	Gallstones Removal	Penicillin	Rash
✎	1111	311	12-JUN-85	John White	15 New Street, New York, N.Y.	Micheal Diamond	Kidney Stones Removal	-	-
✎	5123	145	10-MAY-85	Paul Kosher	Blink Brook Mamaroneck, N.Y.	Beth Little	Gallstones removal	-	-
✎	1234	243	05-APR-84	Mary Jones	10 Main Street, Rye, N.Y.	Charles Field	Eye cataract removal	Tetracycline	Fever
✎	1234	467	10-MAY-85	Mary Jones	10 Main Street, Rye, N.Y.	Patricia Gold	Thrombosis Removal	-	-
✎	2345	189	08-JAN-86	Charles Brown	Dogwood Lane, Harrison, N.Y.	David Rosen	Open heart surgery	Cephalosporin	-
✎	4876	145	05-NOV-85	Hal Kane	55 Boston Post Road, Chester, Conn.	Beth Little	Cholecystectomy	Demicillin	-
✎	6845	243	05-APR-84	Ann Hood	Hilton Road Larchmont, N.Y.	Charles Field	Eye corona replacement	Tetracycline	Fever
✎	6845	243	15-DEC-84	Ann Hood	Hilton Road Larchmont, N.Y.	Charles Field	Eye cataract removal	-	-
									row(s) 1 - 9 of 9

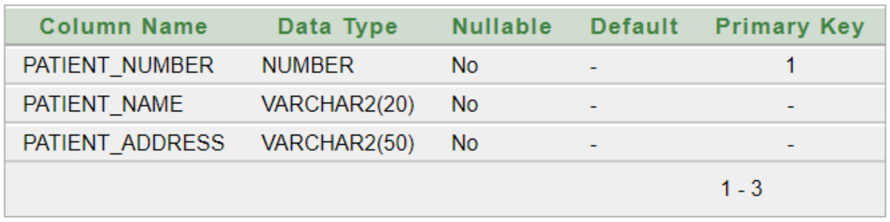
Download

Simplified Hospital Data Table

(2) Functional Dependencies

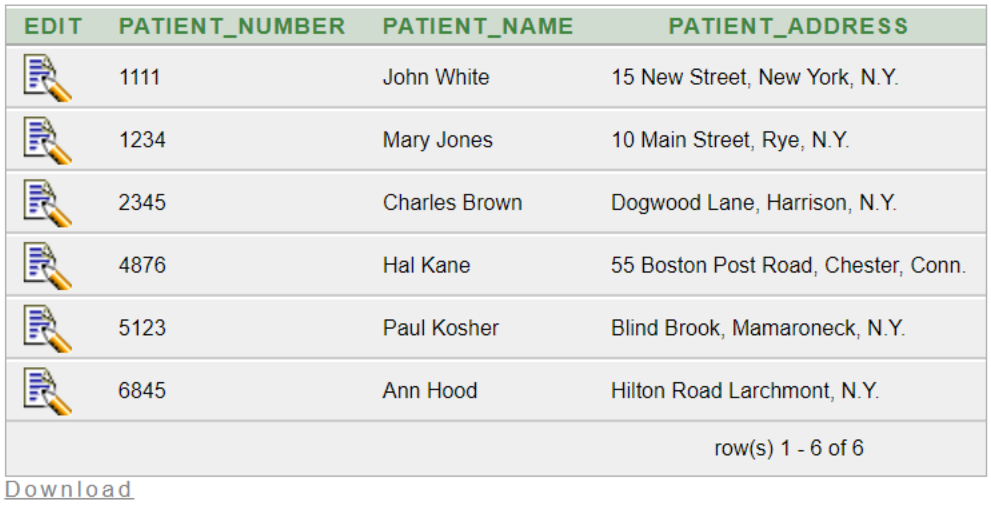
1. (PATIENT_NUMBER, SURGEON_LICENSE_NUMBER, DATE_OF_SURGERY) → (PATIENT_NAME, PATEINT_ADDRESS, SURGEON_NAME, SURGERY, POSTOPERATIVE_DRUG_ADMINISTERED, SIDE_EFFECT_OF_DRUG)
2. (PATIENT_NUMBER) → (PATIENT_NAME, PATIENT_ADDRESS)
3. (SURGEON_LICENSE_NUMBER) → (SURGEON_NAME)
4. (DRUG_ADMINISTERED) → (SIDE_EFFECT_OF_DRUG)

(3) Tables from Dependencies



Column Name	Data Type	Nullable	Default	Primary Key
PATIENT_NUMBER	NUMBER	No	-	1
PATIENT_NAME	VARCHAR2(20)	No	-	-
PATIENT_ADDRESS	VARCHAR2(50)	No	-	-
				1 - 3

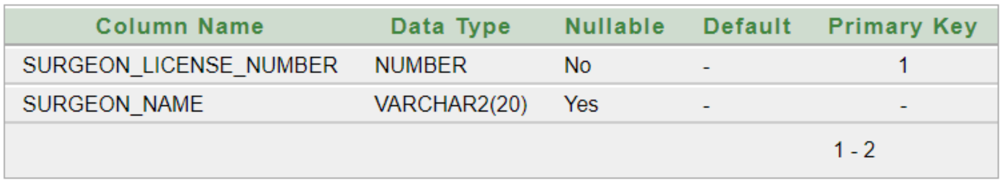
Patient Table Structure



EDIT	PATIENT_NUMBER	PATIENT_NAME	PATIENT_ADDRESS
✎	1111	John White	15 New Street, New York, N.Y.
✎	1234	Mary Jones	10 Main Street, Rye, N.Y.
✎	2345	Charles Brown	Dogwood Lane, Harrison, N.Y.
✎	4876	Hal Kane	55 Boston Post Road, Chester, Conn.
✎	5123	Paul Kosher	Blind Brook, Mamaroneck, N.Y.
✎	6845	Ann Hood	Hilton Road Larchmont, N.Y.
			row(s) 1 - 6 of 6

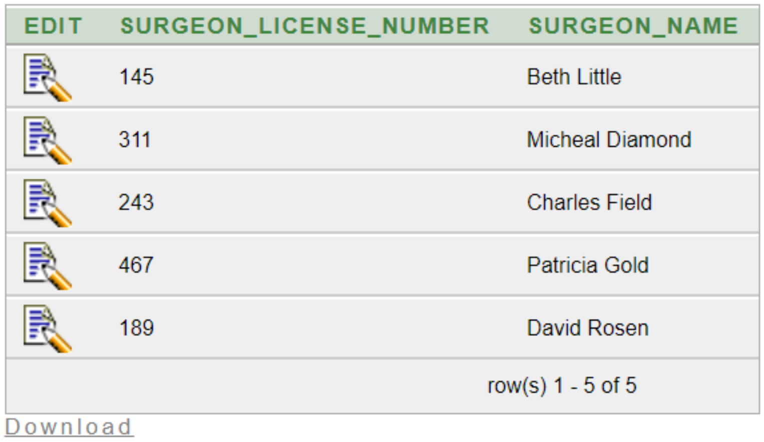
Download

Patient Table Data



Column Name	Data Type	Nullable	Default	Primary Key
SURGEON_LICENSE_NUMBER	NUMBER	No	-	1
SURGEON_NAME	VARCHAR2(20)	Yes	-	-
				1 - 2

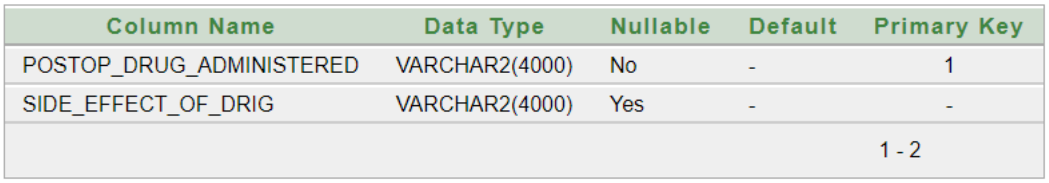
Surgeon Table Structure



EDIT	SURGEON_LICENSE_NUMBER	SURGEON_NAME
✎	145	Beth Little
✎	311	Micheal Diamond
✎	243	Charles Field
✎	467	Patricia Gold
✎	189	David Rosen
		row(s) 1 - 5 of 5

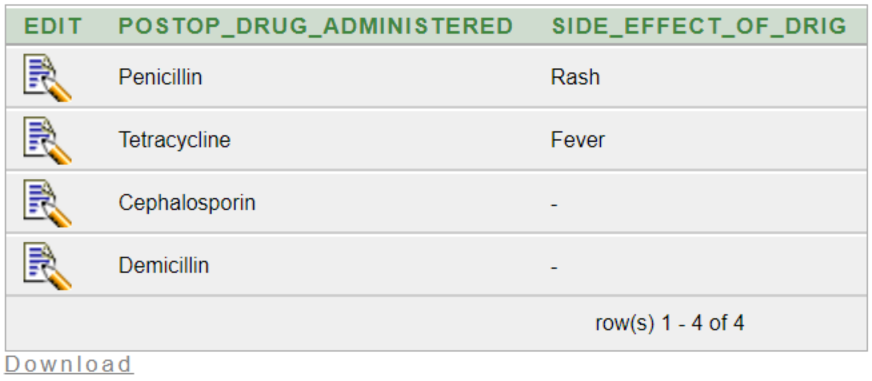
Download

Surgeon Table Data



Column Name	Data Type	Nullable	Default	Primary Key
POSTOP_DRUG_ADMINISTERED	VARCHAR2(4000)	No	-	1
SIDE_EFFECT_OF_DRIG	VARCHAR2(4000)	Yes	-	-
				1 - 2

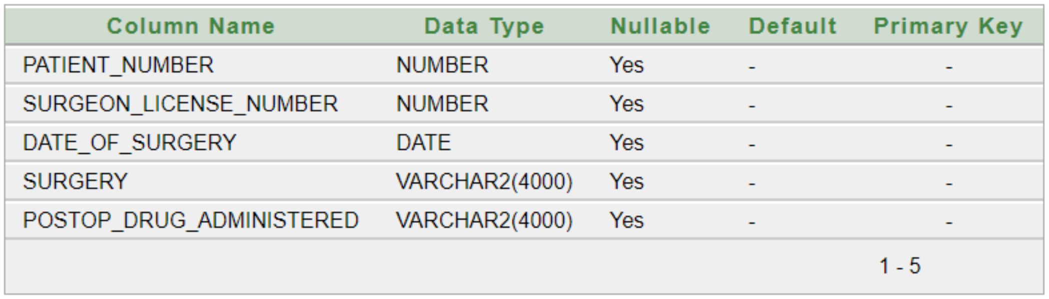
Drug Table Structure



EDIT	POSTOP_DRUG_ADMINISTERED	SIDE_EFFECT_OF_DRIG
	Penicillin	Rash
	Tetracycline	Fever
	Cephalosporin	-
	Demicillin	-
		row(s) 1 - 4 of 4

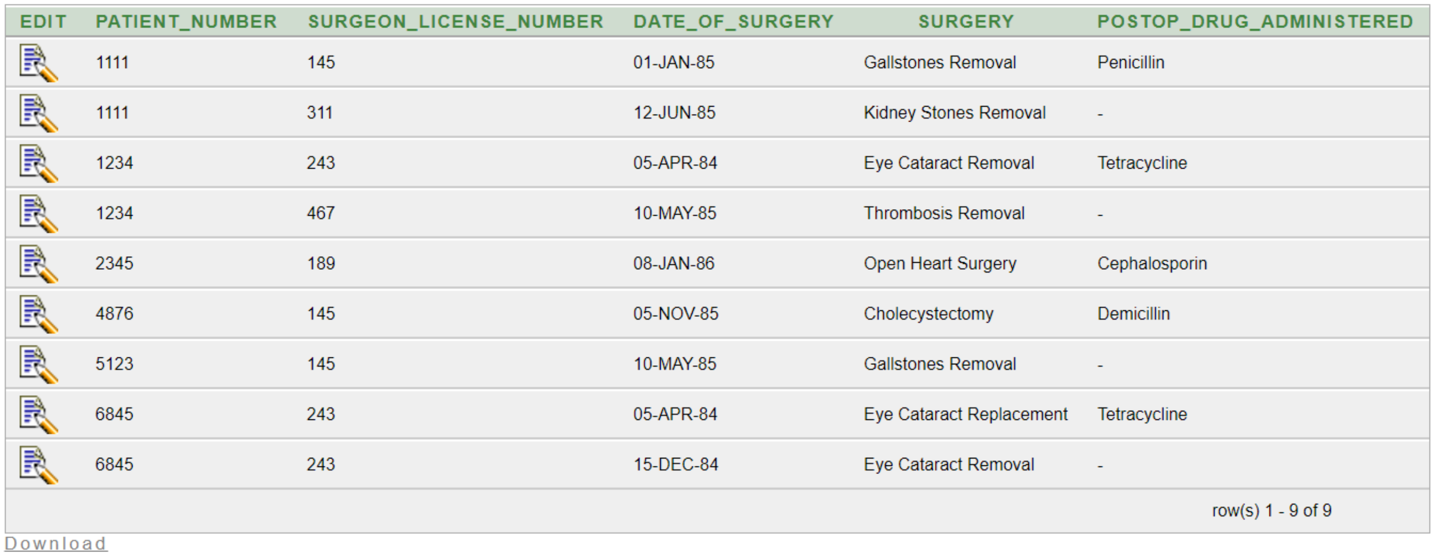
Download

Drug Table Data



Column Name	Data Type	Nullable	Default	Primary Key
PATIENT_NUMBER	NUMBER	Yes	-	-
SURGEON_LICENSE_NUMBER	NUMBER	Yes	-	-
DATE_OF_SURGERY	DATE	Yes	-	-
SURGERY	VARCHAR2(4000)	Yes	-	-
POSTOP_DRUG_ADMINISTERED	VARCHAR2(4000)	Yes	-	-
				1 - 5

Surgery Info Table Structure



EDIT	PATIENT_NUMBER	SURGEON_LICENSE_NUMBER	DATE_OF_SURGERY	SURGERY	POSTOP_DRUG_ADMINISTERED
	1111	145	01-JAN-85	Gallstones Removal	Penicillin
	1111	311	12-JUN-85	Kidney Stones Removal	-
	1234	243	05-APR-84	Eye Cataract Removal	Tetracycline
	1234	467	10-MAY-85	Thrombosis Removal	-
	2345	189	08-JAN-86	Open Heart Surgery	Cephalosporin
	4876	145	05-NOV-85	Cholecystectomy	Demicillin
	5123	145	10-MAY-85	Gallstones Removal	-
	6845	243	05-APR-84	Eye Cataract Replacement	Tetracycline
	6845	243	15-DEC-84	Eye Cataract Removal	-
					row(s) 1 - 9 of 9

Download

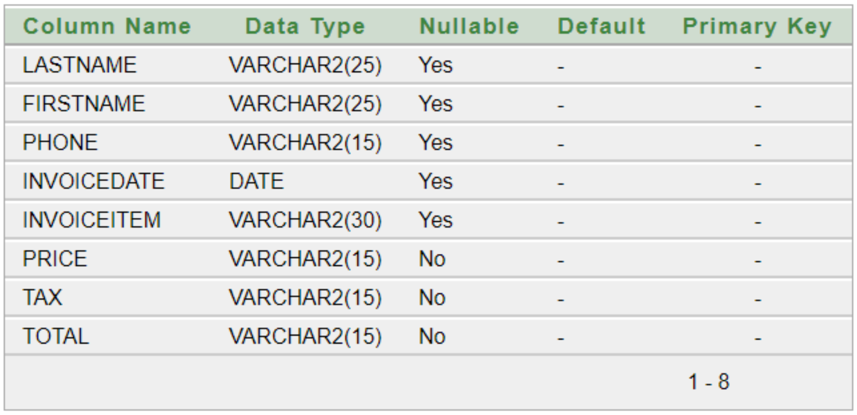
Surgery Info Table Data

Chapter 6 – Queen Anne Curiosity Shop Functional Dependencies and Normalizations

6.1. Project Description

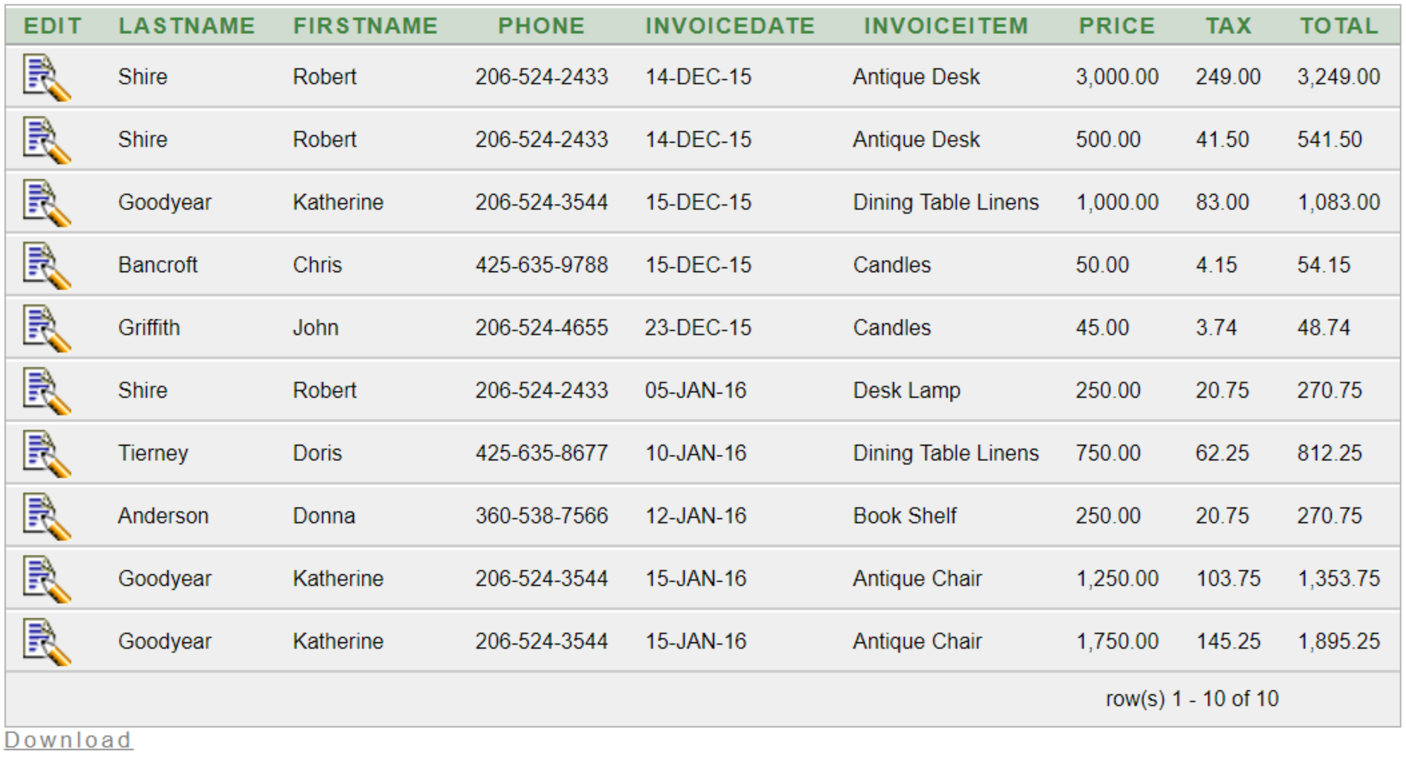
The goal of project 6 was to find and list all the functional dependencies in the provided customer table from the Queen Anne Curiosity Shop, and to then normalize the customer table using the found functional dependencies. The project was completed using Oracle and MySQL. To complete the project the customer table was created using Oracle. The functional dependencies were then derived from observing the table, and the table was decomposed into second normal form tables and those tables were decomposed into third normal form tables. All tables in second normal form are free of partial dependencies and in first normal form. All tables in third normal form are free of transitive dependencies and in second normal form. This project was successful in teaching one how to recognize functional dependencies in any given table and to then normalize those tables.

6.2. Tables/Input/Queries and Associated Output



Column Name	Data Type	Nullable	Default	Primary Key
LASTNAME	VARCHAR2(25)	Yes	-	-
FIRSTNAME	VARCHAR2(25)	Yes	-	-
PHONE	VARCHAR2(15)	Yes	-	-
INVOICEDATE	DATE	Yes	-	-
INVOICEITEM	VARCHAR2(30)	Yes	-	-
PRICE	VARCHAR2(15)	No	-	-
TAX	VARCHAR2(15)	No	-	-
TOTAL	VARCHAR2(15)	No	-	-
				1 - 8

QACS Sales Data Table Structure



EDIT	LASTNAME	FIRSTNAME	PHONE	INVOICEDATE	INVOICEITEM	PRICE	TAX	TOTAL
✎	Shire	Robert	206-524-2433	14-DEC-15	Antique Desk	3,000.00	249.00	3,249.00
✎	Shire	Robert	206-524-2433	14-DEC-15	Antique Desk	500.00	41.50	541.50
✎	Goodyear	Katherine	206-524-3544	15-DEC-15	Dining Table Linens	1,000.00	83.00	1,083.00
✎	Bancroft	Chris	425-635-9788	15-DEC-15	Candles	50.00	4.15	54.15
✎	Griffith	John	206-524-4655	23-DEC-15	Candles	45.00	3.74	48.74
✎	Shire	Robert	206-524-2433	05-JAN-16	Desk Lamp	250.00	20.75	270.75
✎	Tierney	Doris	425-635-8677	10-JAN-16	Dining Table Linens	750.00	62.25	812.25
✎	Anderson	Donna	360-538-7566	12-JAN-16	Book Shelf	250.00	20.75	270.75
✎	Goodyear	Katherine	206-524-3544	15-JAN-16	Antique Chair	1,250.00	103.75	1,353.75
✎	Goodyear	Katherine	206-524-3544	15-JAN-16	Antique Chair	1,750.00	145.25	1,895.25
						row(s) 1 - 10 of 10		

Download

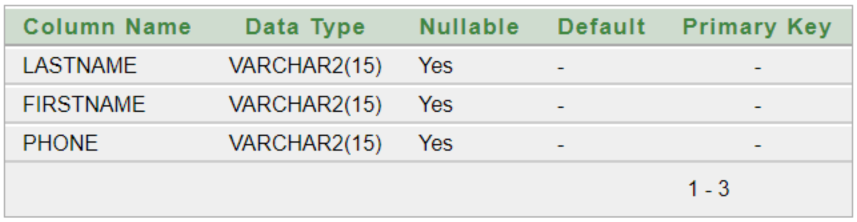
QACS Sales Data Table Data

(1) Functional Dependencies

1. (Price, Tax) → (Total)
2. (LastName, FirstName) → (Phone)
3. (LastName, FirstName, Phone, InvoiceDate, InvoiceItem) → (Price, Tax, Total)
4. (InvoiceDate, InvoiceItem,Phone) → (Price, Tax, Total)

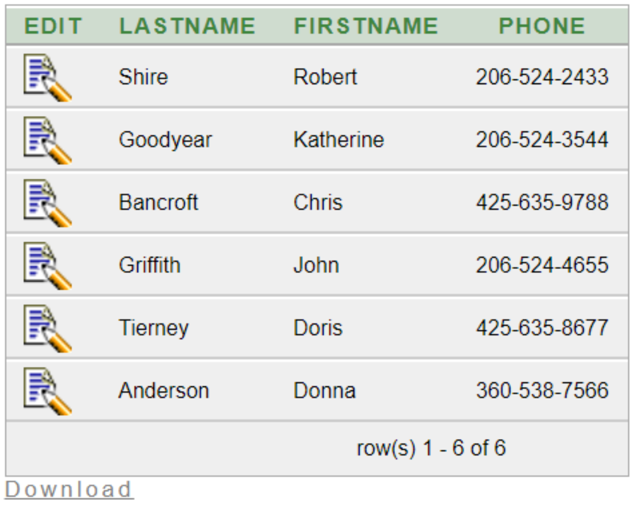
(2) Second Normal Form

Customer Table
PRIMARY KEY: (<u>LastName, FirstName</u>)



Column Name	Data Type	Nullable	Default	Primary Key
LASTNAME	VARCHAR2(15)	Yes	-	-
FIRSTNAME	VARCHAR2(15)	Yes	-	-
PHONE	VARCHAR2(15)	Yes	-	-
				1 - 3

Customer Table Structure

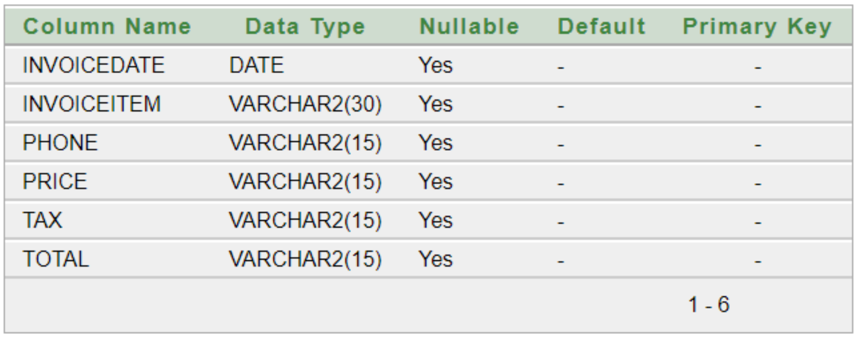


EDIT	LASTNAME	FIRSTNAME	PHONE
✎	Shire	Robert	206-524-2433
✎	Goodyear	Katherine	206-524-3544
✎	Bancroft	Chris	425-635-9788
✎	Griffith	John	206-524-4655
✎	Tierney	Doris	425-635-8677
✎	Anderson	Donna	360-538-7566
		row(s) 1 - 6 of 6	

Download

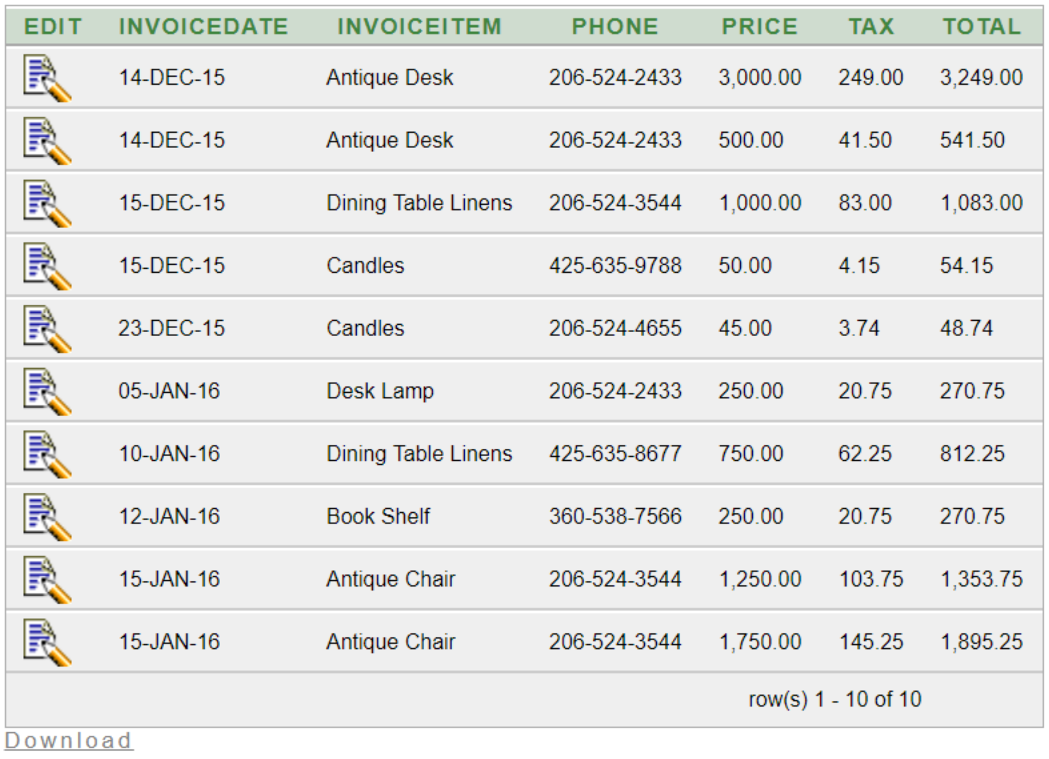
Customer Table Data

Invoice Table
PRIMARY KEY: (<u>InvoiceDate, InvoiceItem, Phone</u>)



Column Name	Data Type	Nullable	Default	Primary Key
INVOICEDATE	DATE	Yes	-	-
INVOICEITEM	VARCHAR2(30)	Yes	-	-
PHONE	VARCHAR2(15)	Yes	-	-
PRICE	VARCHAR2(15)	Yes	-	-
TAX	VARCHAR2(15)	Yes	-	-
TOTAL	VARCHAR2(15)	Yes	-	-
				1 - 6

Invoice Table Structure



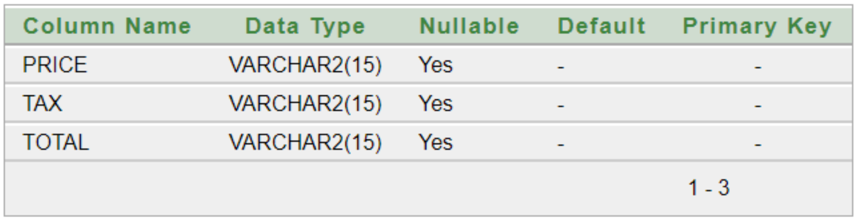
EDIT	INVOICEDATE	INVOICEITEM	PHONE	PRICE	TAX	TOTAL
✎	14-DEC-15	Antique Desk	206-524-2433	3,000.00	249.00	3,249.00
✎	14-DEC-15	Antique Desk	206-524-2433	500.00	41.50	541.50
✎	15-DEC-15	Dining Table Linens	206-524-3544	1,000.00	83.00	1,083.00
✎	15-DEC-15	Candles	425-635-9788	50.00	4.15	54.15
✎	23-DEC-15	Candles	206-524-4655	45.00	3.74	48.74
✎	05-JAN-16	Desk Lamp	206-524-2433	250.00	20.75	270.75
✎	10-JAN-16	Dining Table Linens	425-635-8677	750.00	62.25	812.25
✎	12-JAN-16	Book Shelf	360-538-7566	250.00	20.75	270.75
✎	15-JAN-16	Antique Chair	206-524-3544	1,250.00	103.75	1,353.75
✎	15-JAN-16	Antique Chair	206-524-3544	1,750.00	145.25	1,895.25

row(s) 1 - 10 of 10

Invoice Table Data

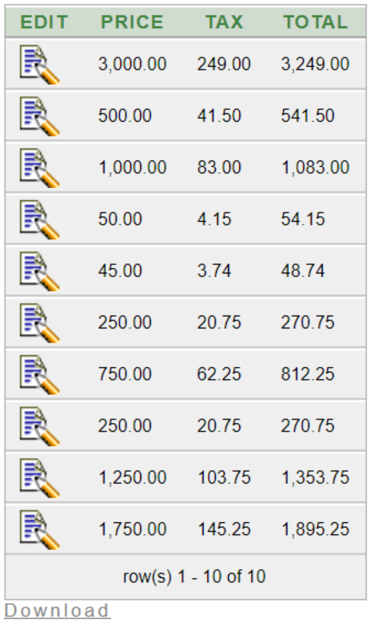
Price Table

PRIMARY KEY: <u>(Price, Tax)</u>



Column Name	Data Type	Nullable	Default	Primary Key
PRICE	VARCHAR2(15)	Yes	-	-
TAX	VARCHAR2(15)	Yes	-	-
TOTAL	VARCHAR2(15)	Yes	-	-
				1 - 3

Price Table Structure



EDIT	PRICE	TAX	TOTAL
✎	3,000.00	249.00	3,249.00
✎	500.00	41.50	541.50
✎	1,000.00	83.00	1,083.00
✎	50.00	4.15	54.15
✎	45.00	3.74	48.74
✎	250.00	20.75	270.75
✎	750.00	62.25	812.25
✎	250.00	20.75	270.75
✎	1,250.00	103.75	1,353.75
✎	1,750.00	145.25	1,895.25

row(s) 1 - 10 of 10

Download

Price Table Data

Chapter 7 – Hospital Data Table Functional Dependencies and Normalization

7.1. Project Description

The goal of project 7 was to find and list all the functional dependencies in the provided simplified hospital data table, and to then normalize the customer table using the found functional dependencies. The project was completed using Oracle and MySQL. To complete the project the simplified hospital data table was created using Oracle. The functional dependencies were then derived from observing the table, and the table was decomposed into second normal form tables and those tables were decomposed into third normal form tables. All tables in second normal form are free of partial dependencies and in first normal form. All tables in third normal form are free of transitive dependencies and in second normal form. An E/R model was also created which allows for one to visually see the relations among tables. This project was successful in teaching one how to recognize functional dependencies in any given table and to then normalize those tables.

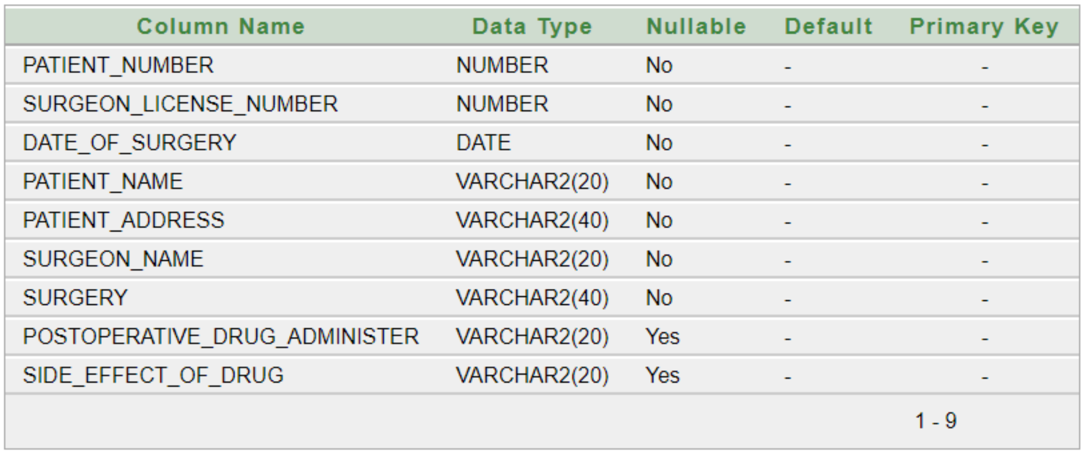
7.2. Tables/Input/Queries and Associated Output

(1) Functional Dependencies

1. (PATIENT_NUMBER, SURGEON_LICENSE_NUMBER, DATE_OF_SURGERY) → (PATIENT_NAME, PATEINT_ADDRESS, SURGEON_NAME, SURGERY, POSTOPERATIVE_DRUG_ADMINISTERED, SIDE_EFFECT_OF_DRUG)
2. (PATIENT_NUMBER) → (PATIENT_NAME, PATIENT_ADDRESS)
3. (SURGEON_LICENSE_NUMBER) → (SURGEON_NAME)
4. (DRUG_ADMINISTERED) → (SIDE_EFFECT_OF_DRUG)

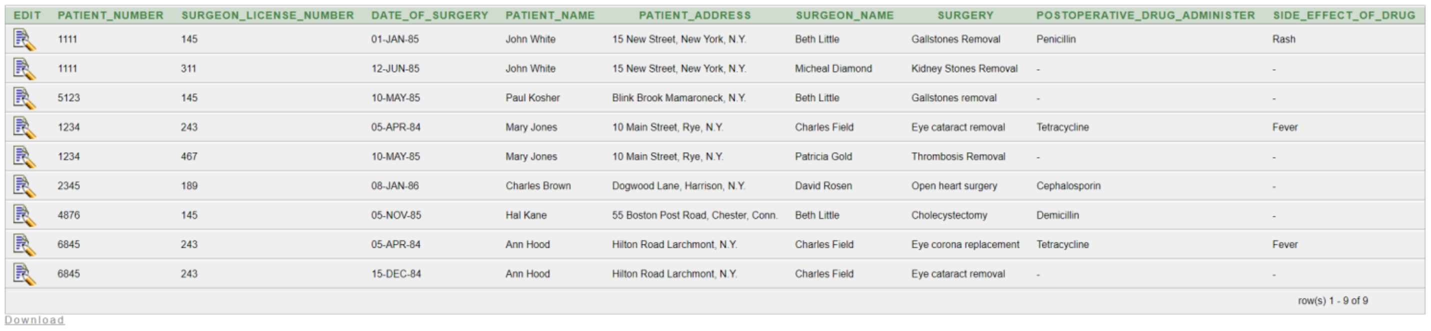
(2) First Normal Form Table

Primary Key: (PATIENT_NUMBER, SURGEON_LICENSE_NUMBER, DATE_OF_SURGERY)



Column Name	Data Type	Nullable	Default	Primary Key
PATIENT_NUMBER	NUMBER	No	-	-
SURGEON_LICENSE_NUMBER	NUMBER	No	-	-
DATE_OF_SURGERY	DATE	No	-	-
PATIENT_NAME	VARCHAR2(20)	No	-	-
PATIENT_ADDRESS	VARCHAR2(40)	No	-	-
SURGEON_NAME	VARCHAR2(20)	No	-	-
SURGERY	VARCHAR2(40)	No	-	-
POSTOPERATIVE_DRUG_ADMINISTER	VARCHAR2(20)	Yes	-	-
SIDE_EFFECT_OF_DRUG	VARCHAR2(20)	Yes	-	-
				1 - 9

Simplified Hospital Data Structure



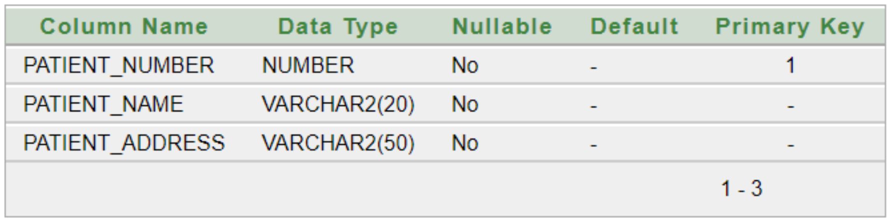
EDIT	PATIENT_NUMBER	SURGEON_LICENSE_NUMBER	DATE_OF_SURGERY	PATIENT_NAME	PATIENT_ADDRESS	SURGEON_NAME	SURGERY	POSTOPERATIVE_DRUG_ADMINISTER	SIDE_EFFECT_OF_DRUG
✎	1111	145	01-JAN-85	John White	15 New Street, New York, N.Y.	Beth Little	Gallstones Removal	Penicillin	Rash
✎	1111	311	12-JUN-85	John White	15 New Street, New York, N.Y.	Micheal Diamond	Kidney Stones Removal	-	-
✎	5123	145	10-MAY-85	Paul Kosher	Blink Brook Mamaroneck, N.Y.	Beth Little	Gallstones removal	-	-
✎	1234	243	05-APR-84	Mary Jones	10 Main Street, Rye, N.Y.	Charles Field	Eye cataract removal	Tetracycline	Fever
✎	1234	467	10-MAY-85	Mary Jones	10 Main Street, Rye, N.Y.	Patricia Gold	Thrombosis Removal	-	-
✎	2345	189	08-JAN-86	Charles Brown	Dogwood Lane, Harrison, N.Y.	David Rosen	Open heart surgery	Cephalosporin	-
✎	4876	145	05-NOV-85	Hal Kane	55 Boston Post Road, Chester, Conn.	Beth Little	Cholecystectomy	Demicillin	-
✎	6845	243	05-APR-84	Ann Hood	Hilton Road Larchmont, N.Y.	Charles Field	Eye corona replacement	Tetracycline	Fever
✎	6845	243	15-DEC-84	Ann Hood	Hilton Road Larchmont, N.Y.	Charles Field	Eye cataract removal	-	-
									row(s) 1 - 9 of 9

Download

Simplified Hospital Data Table

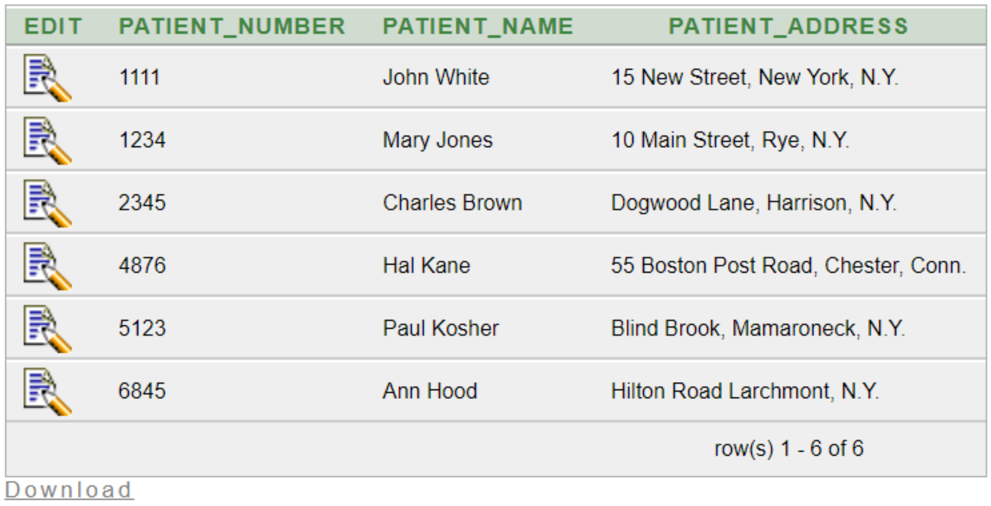
(3) Second Normal Form Tables

Patient Table
Primary Key: (PATIENT_NUMBER)



Column Name	Data Type	Nullable	Default	Primary Key
PATIENT_NUMBER	NUMBER	No	-	1
PATIENT_NAME	VARCHAR2(20)	No	-	-
PATIENT_ADDRESS	VARCHAR2(50)	No	-	-
				1 - 3

Patient Table Structure

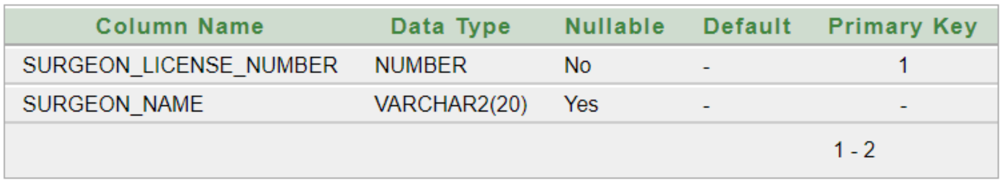


EDIT	PATIENT_NUMBER	PATIENT_NAME	PATIENT_ADDRESS
✎	1111	John White	15 New Street, New York, N.Y.
✎	1234	Mary Jones	10 Main Street, Rye, N.Y.
✎	2345	Charles Brown	Dogwood Lane, Harrison, N.Y.
✎	4876	Hal Kane	55 Boston Post Road, Chester, Conn.
✎	5123	Paul Kosher	Blind Brook, Mamaroneck, N.Y.
✎	6845	Ann Hood	Hilton Road Larchmont, N.Y.
			row(s) 1 - 6 of 6

Download

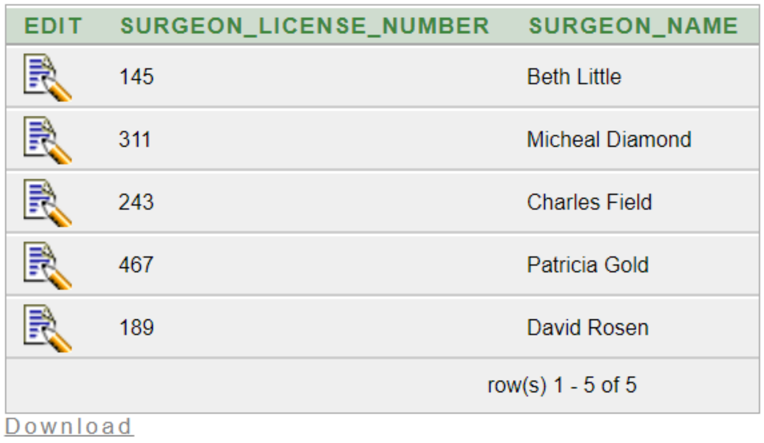
Patient Table Data

Surgeon Table
Primary Key: (SURGEON_LICENSE_NUMBER)



Column Name	Data Type	Nullable	Default	Primary Key
SURGEON_LICENSE_NUMBER	NUMBER	No	-	1
SURGEON_NAME	VARCHAR2(20)	Yes	-	-
				1 - 2

Surgeon Table Structure

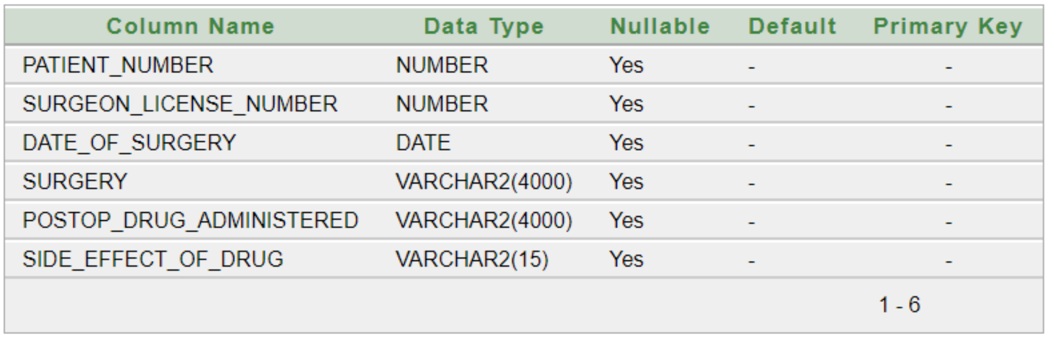


EDIT	SURGEON_LICENSE_NUMBER	SURGEON_NAME
	145	Beth Little
	311	Micheal Diamond
	243	Charles Field
	467	Patricia Gold
	189	David Rosen
		row(s) 1 - 5 of 5

Download

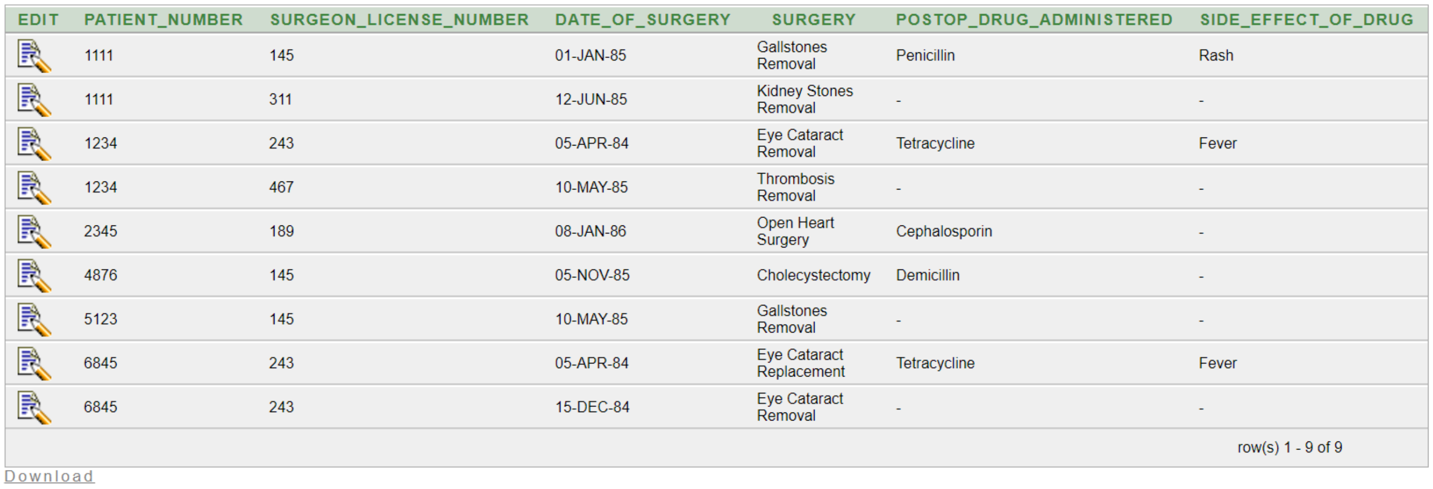
Surgeon Table Data

Surgery Info Table
Primary Key: (PATIENT_NUMBER, SURGEON_LICENSE_NUMBER, DATE_OF_SURGERY)



Column Name	Data Type	Nullable	Default	Primary Key
PATIENT_NUMBER	NUMBER	Yes	-	-
SURGEON_LICENSE_NUMBER	NUMBER	Yes	-	-
DATE_OF_SURGERY	DATE	Yes	-	-
SURGERY	VARCHAR2(4000)	Yes	-	-
POSTOP_DRUG_ADMINISTERED	VARCHAR2(4000)	Yes	-	-
SIDE_EFFECT_OF_DRUG	VARCHAR2(15)	Yes	-	-
				1 - 6

Surgery Info Table Structure



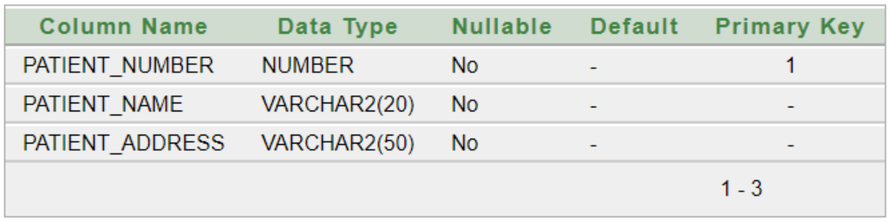
EDIT	PATIENT_NUMBER	SURGEON_LICENSE_NUMBER	DATE_OF_SURGERY	SURGERY	POSTOP_DRUG_ADMINISTERED	SIDE_EFFECT_OF_DRUG
✎	1111	145	01-JAN-85	Gallstones Removal	Penicillin	Rash
✎	1111	311	12-JUN-85	Kidney Stones Removal	-	-
✎	1234	243	05-APR-84	Eye Cataract Removal	Tetracycline	Fever
✎	1234	467	10-MAY-85	Thrombosis Removal	-	-
✎	2345	189	08-JAN-86	Open Heart Surgery	Cephalosporin	-
✎	4876	145	05-NOV-85	Cholecystectomy	Demicillin	-
✎	5123	145	10-MAY-85	Gallstones Removal	-	-
✎	6845	243	05-APR-84	Eye Cataract Replacement	Tetracycline	Fever
✎	6845	243	15-DEC-84	Eye Cataract Removal	-	-
						row(s) 1 - 9 of 9

Download

Surgery Info Table Data

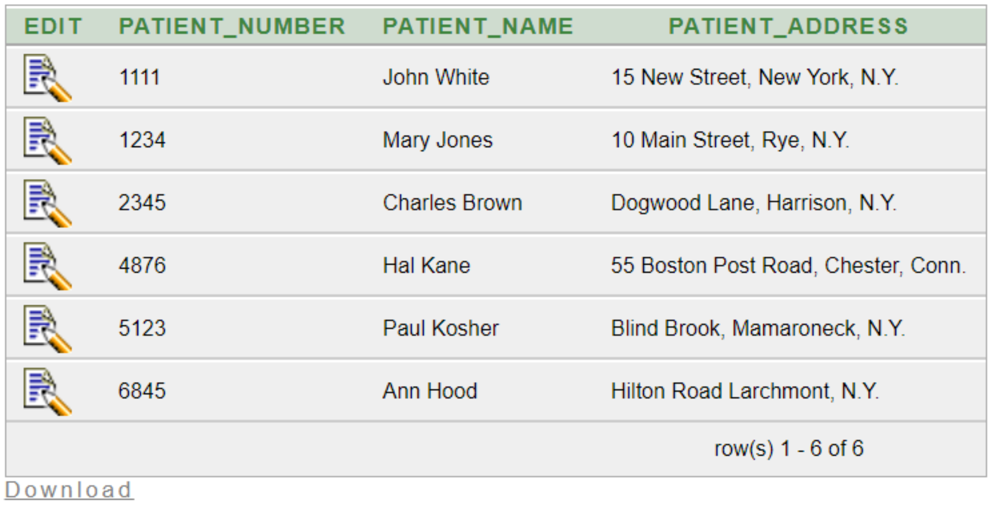
(4) Third Normal Form Tables

Patient Table
Primary Key: (PATIENT_NUMBER)



Column Name	Data Type	Nullable	Default	Primary Key
PATIENT_NUMBER	NUMBER	No	-	1
PATIENT_NAME	VARCHAR2(20)	No	-	-
PATIENT_ADDRESS	VARCHAR2(50)	No	-	-
				1 - 3

Patient Table Structure

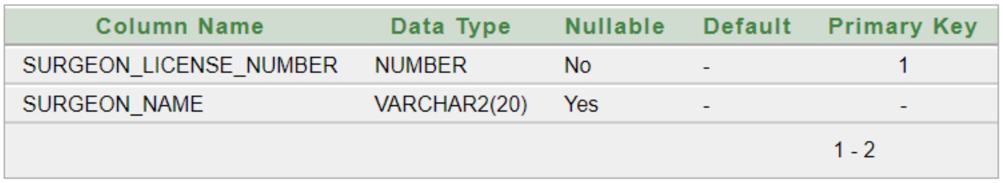


EDIT	PATIENT_NUMBER	PATIENT_NAME	PATIENT_ADDRESS
✎	1111	John White	15 New Street, New York, N.Y.
✎	1234	Mary Jones	10 Main Street, Rye, N.Y.
✎	2345	Charles Brown	Dogwood Lane, Harrison, N.Y.
✎	4876	Hal Kane	55 Boston Post Road, Chester, Conn.
✎	5123	Paul Kosher	Blind Brook, Mamaroneck, N.Y.
✎	6845	Ann Hood	Hilton Road Larchmont, N.Y.
			row(s) 1 - 6 of 6

Download

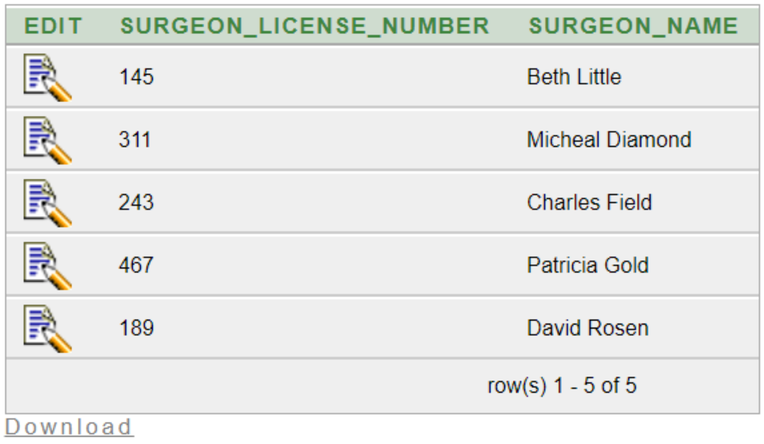
Patient Table Data

Surgeon Table
Primary Key: (SURGEON_LICENSE_NUMBER)



Column Name	Data Type	Nullable	Default	Primary Key
SURGEON_LICENSE_NUMBER	NUMBER	No	-	1
SURGEON_NAME	VARCHAR2(20)	Yes	-	-
				1 - 2

Surgeon Table Structure

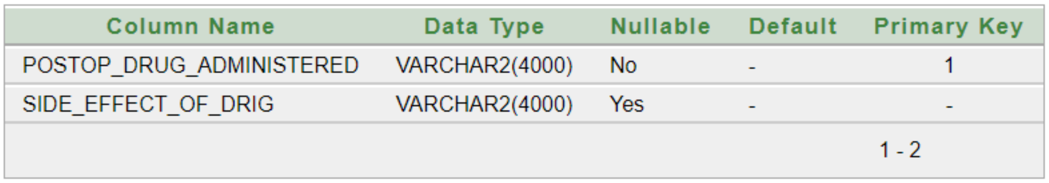


EDIT	SURGEON_LICENSE_NUMBER	SURGEON_NAME
✎	145	Beth Little
✎	311	Micheal Diamond
✎	243	Charles Field
✎	467	Patricia Gold
✎	189	David Rosen
		row(s) 1 - 5 of 5

Download

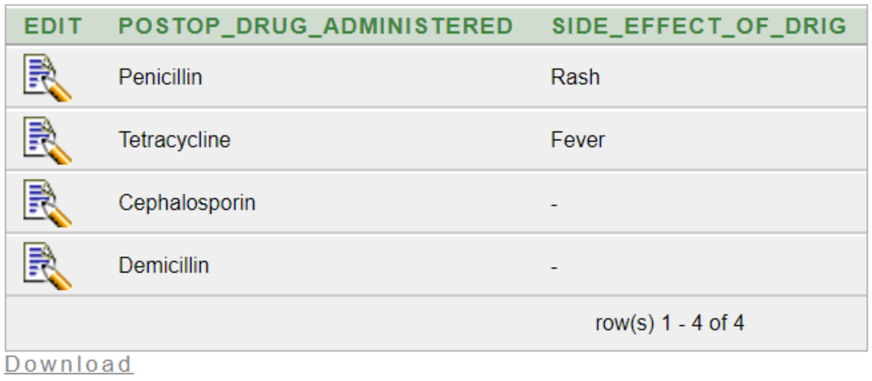
Surgeon Table Data

Drug Table
Primary Key: (POSTOP_DRUG_ADMINISTERD)



Column Name	Data Type	Nullable	Default	Primary Key
POSTOP_DRUG_ADMINISTERED	VARCHAR2(4000)	No	-	1
SIDE_EFFECT_OF_DRIG	VARCHAR2(4000)	Yes	-	-
				1 - 2

Drug Table Structure

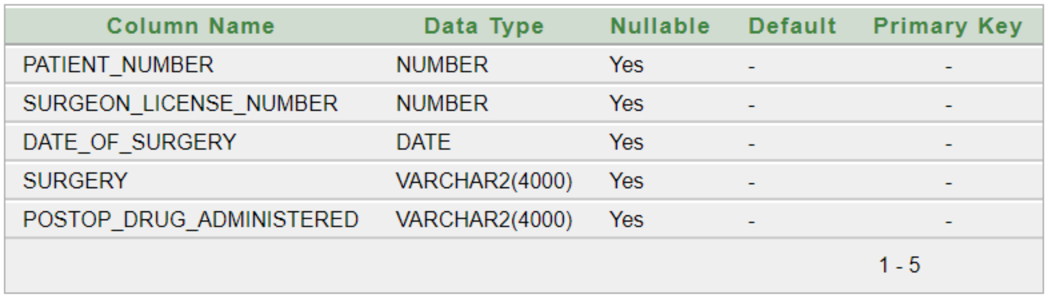


EDIT	POSTOP_DRUG_ADMINISTERED	SIDE_EFFECT_OF_DRIG
	Penicillin	Rash
	Tetracycline	Fever
	Cephalosporin	-
	Demicillin	-
		row(s) 1 - 4 of 4

Download

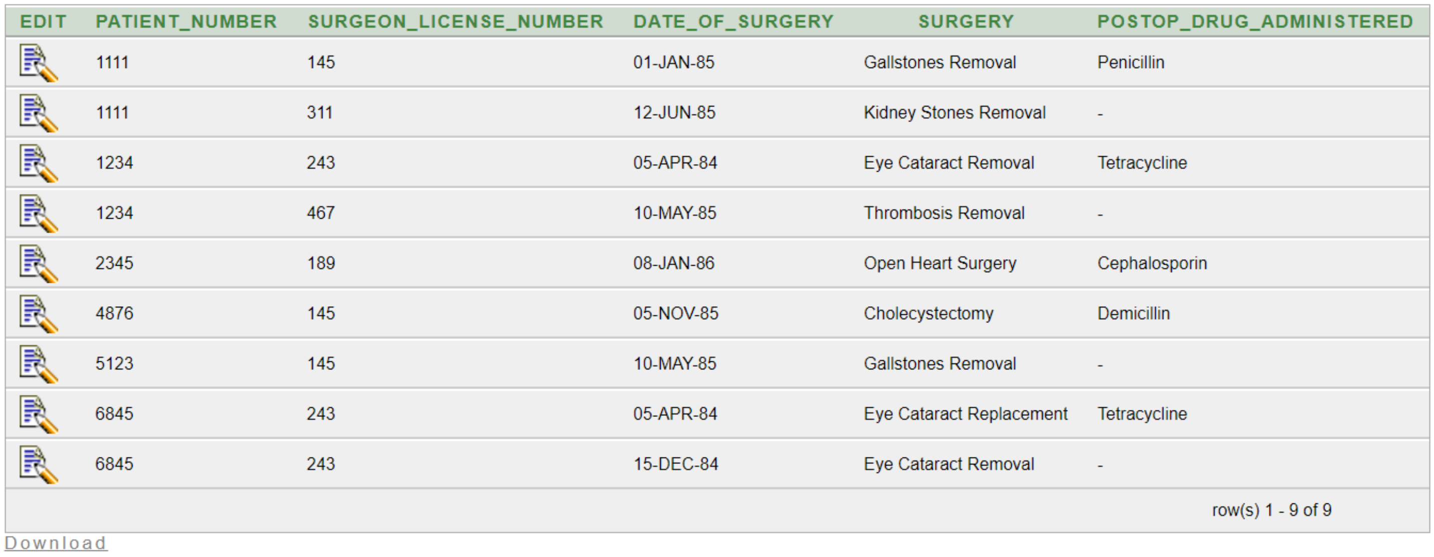
Drug Table Data

Surgery Info Table
Primary Key: (PATIENT_NUMBER, SURGEON_LICENSE_NUMBER, DATE_OF_SURGERY)



Column Name	Data Type	Nullable	Default	Primary Key
PATIENT_NUMBER	NUMBER	Yes	-	-
SURGEON_LICENSE_NUMBER	NUMBER	Yes	-	-
DATE_OF_SURGERY	DATE	Yes	-	-
SURGERY	VARCHAR2(4000)	Yes	-	-
POSTOP_DRUG_ADMINISTERED	VARCHAR2(4000)	Yes	-	-
				1 - 5

Surgery Info Table Structure

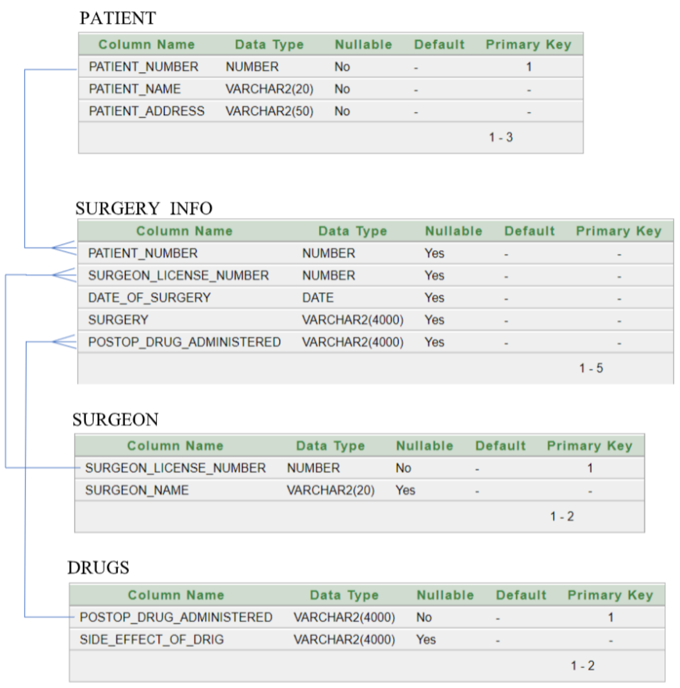


EDIT	PATIENT_NUMBER	SURGEON_LICENSE_NUMBER	DATE_OF_SURGERY	SURGERY	POSTOP_DRUG_ADMINISTERED
✎	1111	145	01-JAN-85	Gallstones Removal	Penicillin
✎	1111	311	12-JUN-85	Kidney Stones Removal	-
✎	1234	243	05-APR-84	Eye Cataract Removal	Tetracycline
✎	1234	467	10-MAY-85	Thrombosis Removal	-
✎	2345	189	08-JAN-86	Open Heart Surgery	Cephalosporin
✎	4876	145	05-NOV-85	Cholecystectomy	Demicillin
✎	5123	145	10-MAY-85	Gallstones Removal	-
✎	6845	243	05-APR-84	Eye Cataract Replacement	Tetracycline
✎	6845	243	15-DEC-84	Eye Cataract Removal	-
					row(s) 1 - 9 of 9

Download

Surgery Info Table Data

(5) E/R Model



PATIENT

Column Name	Data Type	Nullable	Default	Primary Key
PATIENT_NUMBER	NUMBER	No	-	1
PATIENT_NAME	VARCHAR2(20)	No	-	-
PATIENT_ADDRESS	VARCHAR2(50)	No	-	-
				1 - 3

SURGERY INFO

Column Name	Data Type	Nullable	Default	Primary Key
PATIENT_NUMBER	NUMBER	Yes	-	-
SURGEON_LICENSE_NUMBER	NUMBER	Yes	-	-
DATE_OF_SURGERY	DATE	Yes	-	-
SURGERY	VARCHAR2(4000)	Yes	-	-
POSTOP_DRUG_ADMINISTERED	VARCHAR2(4000)	Yes	-	-
				1 - 5

SURGEON

Column Name	Data Type	Nullable	Default	Primary Key
SURGEON_LICENSE_NUMBER	NUMBER	No	-	1
SURGEON_NAME	VARCHAR2(20)	Yes	-	-
				1 - 2

DRUGS

Column Name	Data Type	Nullable	Default	Primary Key
POSTOP_DRUG_ADMINISTERED	VARCHAR2(4000)	No	-	1
SIDE_EFFECT_OF_DRIG	VARCHAR2(4000)	Yes	-	-
				1 - 2

Chapter 8 – Hospital Data Table Queries

8.1. Project Description

The goal of project 8 was to query the third normal form tables created to represent the data in the provided simplified hospital table, from the previous project. The project was completed using Oracle and MySQL.
To complete the project the tables in third normal form were taken from the previous project and displayed along with the functional dependencies found in the tables. All tables in third normal form are free of transitive dependencies and in second normal form. The tables relations were double checked for accuracy to ensure that queries on the tables could be performed properly. An E/R model was created which allows for one to visually see the relations among tables. Ten (10) unique queries were then written in SQL to pull data from the tables. The tables, all queries, and the results, along with a plain English explanation of each query, may be seen in the section below. This project was successful in teaching one how to relate multiple tables and then carry out unique queries on those tables.

8.2. Tables/Input/Queries and Associated Output

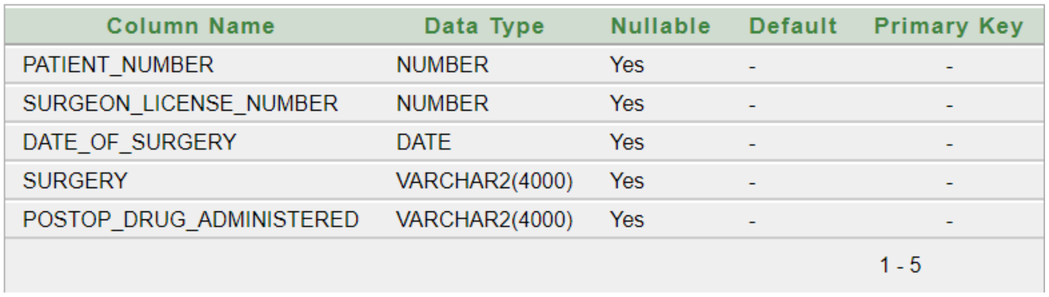
Part A:

Surgery_Info:

Create Statement:

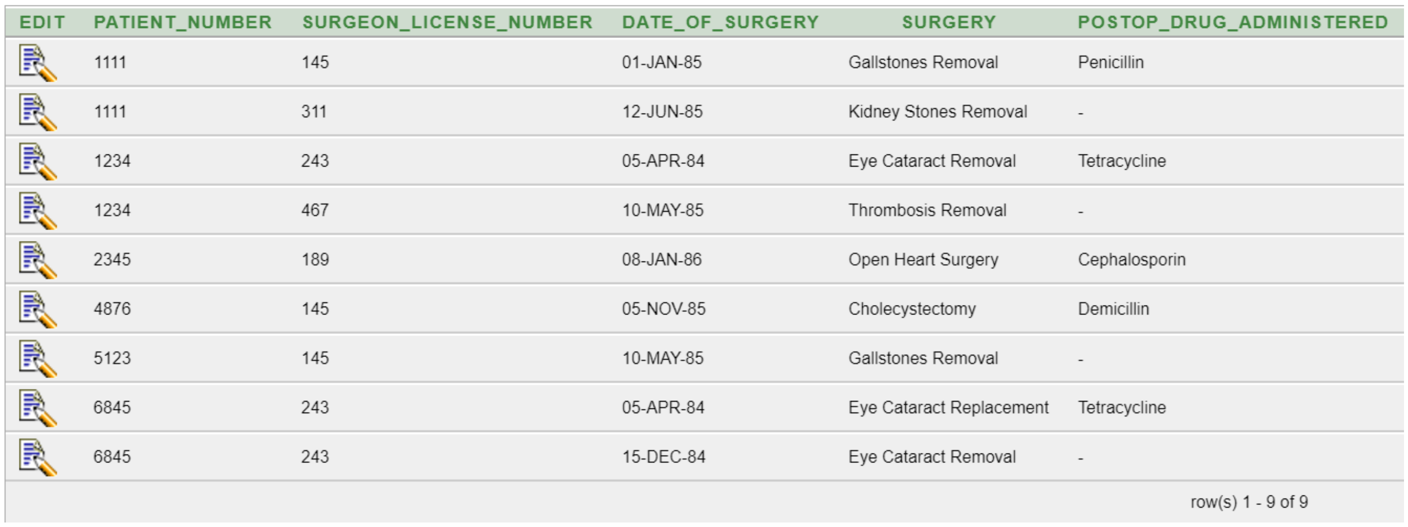
```
CREATE TABLE  "SURGERY_INFO"
   (     "PATIENT_NUMBER" NUMBER,
         "SURGEON_LICENSE_NUMBER" NUMBER,
         "DATE_OF_SURGERY" DATE,
         "SURGERY" VARCHAR2(4000),
         "POSTOP_DRUG_ADMINISTERED" VARCHAR2(4000),
          CONSTRAINT "SURGERY_INFO_FK1" FOREIGN KEY ("PATIENT_NUMBER")
           REFERENCES  "PATIENT" ("PATIENT_NUMBER") ENABLE,
          CONSTRAINT "SURGERY_INFO_FK2" FOREIGN KEY ("SURGEON_LICENSE_NUMBER")
           REFERENCES  "SURGEON" ("SURGEON_LICENSE_NUMBER") ENABLE,
          CONSTRAINT "SURGERY_INFO_FK3" FOREIGN KEY ("POSTOP_DRUG_ADMINISTERED")
           REFERENCES  "DRUGS" ("POSTOP_DRUG_ADMINISTERED") ENABLE
   )
```

Functional Dependency:
(PATIENT_NUMBER, SURGEON_LICENSE_NUMBER, DATE_OF_SURGERY) → (PATIENT_NAME, PATEINT_ADDRESS, SURGEON_NAME, SURGERY, POSTOPERATIVE_DRUG_ADMINISTERED, SIDE_EFFECT_OF_DRUG)



Column Name	Data Type	Nullable	Default	Primary Key
PATIENT_NUMBER	NUMBER	Yes	-	-
SURGEON_LICENSE_NUMBER	NUMBER	Yes	-	-
DATE_OF_SURGERY	DATE	Yes	-	-
SURGERY	VARCHAR2(4000)	Yes	-	-
POSTOP_DRUG_ADMINISTERED	VARCHAR2(4000)	Yes	-	-
				1 - 5

Surgery_Info Table Structure



EDIT	PATIENT_NUMBER	SURGEON_LICENSE_NUMBER	DATE_OF_SURGERY	SURGERY	POSTOP_DRUG_ADMINISTERED
	1111	145	01-JAN-85	Gallstones Removal	Penicillin
	1111	311	12-JUN-85	Kidney Stones Removal	-
	1234	243	05-APR-84	Eye Cataract Removal	Tetracycline
	1234	467	10-MAY-85	Thrombosis Removal	-
	2345	189	08-JAN-86	Open Heart Surgery	Cephalosporin
	4876	145	05-NOV-85	Cholecystectomy	Demicillin
	5123	145	10-MAY-85	Gallstones Removal	-
	6845	243	05-APR-84	Eye Cataract Replacement	Tetracycline
	6845	243	15-DEC-84	Eye Cataract Removal	-

row(s) 1 - 9 of 9

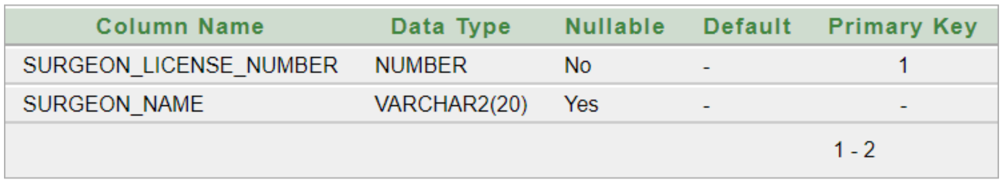
Surgery_Info Table Data

Surgeon:

Create Statement:

```
CREATE TABLE  "SURGEON"
   (	"SURGEON_LICENSE_NUMBER" NUMBER,
	"SURGEON_NAME" VARCHAR2(20),
	 CONSTRAINT "SURGEON_PK" PRIMARY KEY ("SURGEON_LICENSE_NUMBER") ENABLE
   )
/

CREATE OR REPLACE TRIGGER  "BI_SURGEON"
  before insert on "SURGEON"
  for each row
begin
    select "SURGEON_SEQ".nextval into :NEW.SURGEON_LICENSE_NUMBER from dual;
end;

/
ALTER TRIGGER  "BI_SURGEON" ENABLE
/
```

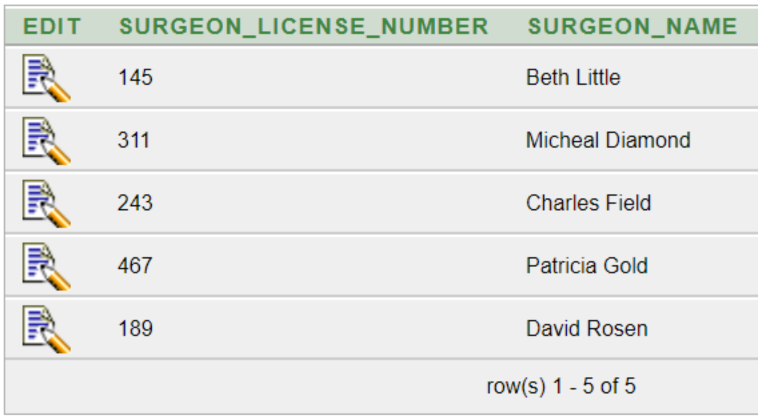
Functional Dependency:
(SURGEON_LICENSE_NUMBER) → (SURGEON_NAME)



Column Name	Data Type	Nullable	Default	Primary Key
SURGEON_LICENSE_NUMBER	NUMBER	No	-	1
SURGEON_NAME	VARCHAR2(20)	Yes	-	-

1 - 2

Surgeon Table Structure



EDIT	SURGEON_LICENSE_NUMBER	SURGEON_NAME
	145	Beth Little
	311	Micheal Diamond
	243	Charles Field
	467	Patricia Gold
	189	David Rosen
		row(s) 1 - 5 of 5

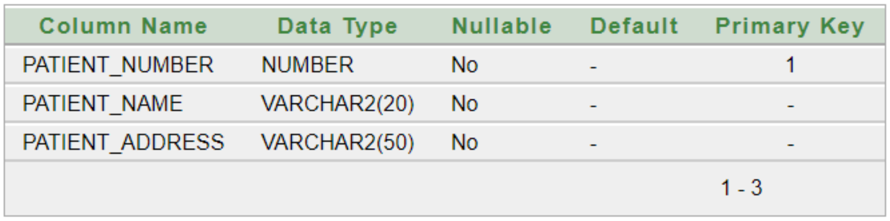
Surgeon Table Data

Patient:

Create Statement:

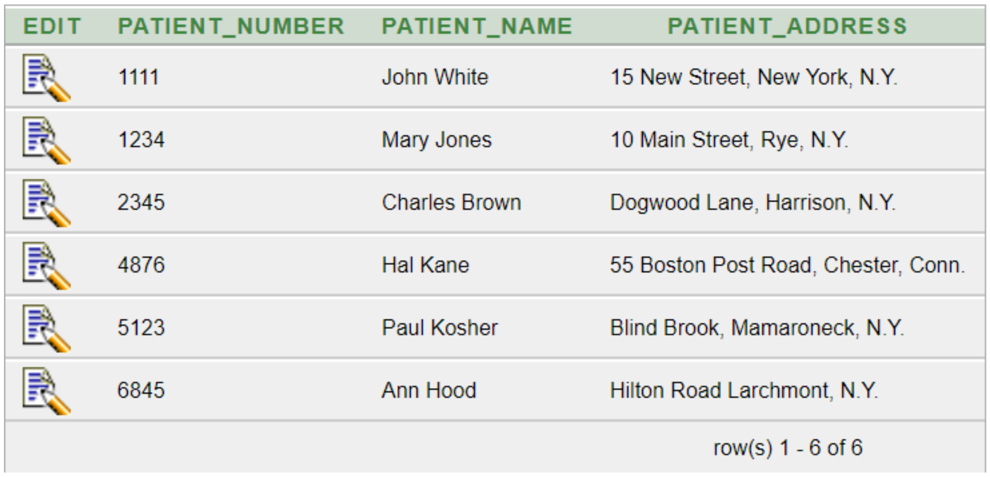
```
CREATE TABLE  "PATIENT"
   (      "PATIENT_NUMBER" NUMBER NOT NULL ENABLE,
          "PATIENT_NAME" VARCHAR2(20) NOT NULL ENABLE,
          "PATIENT_ADDRESS" VARCHAR2(50) NOT NULL ENABLE,
           CONSTRAINT "PATIENT_PK" PRIMARY KEY ("PATIENT_NUMBER") ENABLE
   )
/

CREATE OR REPLACE TRIGGER  "BI_PATIENT"
  before insert on "PATIENT"
  for each row
begin
    select "PATIENT_SEQ".nextval into :NEW.PATIENT_NUMBER from dual;
end;

/
ALTER TRIGGER  "BI_PATIENT" ENABLE
/
```

Functional Dependency:
(PATIENT_NUMBER) → (PATIENT_NAME, PATIENT_ADDRESS)



Column Name	Data Type	Nullable	Default	Primary Key
PATIENT_NUMBER	NUMBER	No	-	1
PATIENT_NAME	VARCHAR2(20)	No	-	-
PATIENT_ADDRESS	VARCHAR2(50)	No	-	-
				1 - 3

Patient Table Structure



EDIT	PATIENT_NUMBER	PATIENT_NAME	PATIENT_ADDRESS
✎	1111	John White	15 New Street, New York, N.Y.
✎	1234	Mary Jones	10 Main Street, Rye, N.Y.
✎	2345	Charles Brown	Dogwood Lane, Harrison, N.Y.
✎	4876	Hal Kane	55 Boston Post Road, Chester, Conn.
✎	5123	Paul Kosher	Blind Brook, Mamaroneck, N.Y.
✎	6845	Ann Hood	Hilton Road Larchmont, N.Y.
			row(s) 1 - 6 of 6

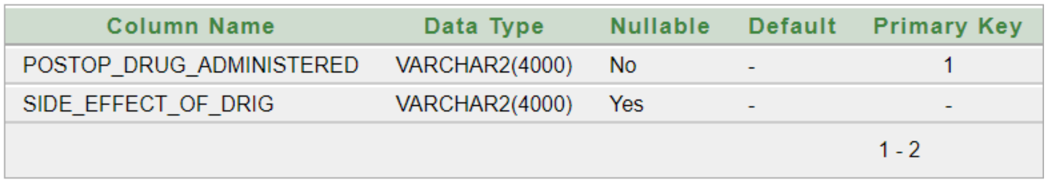
Patient Table Data

<u>Drugs:</u>

<u>Create Statement:</u>

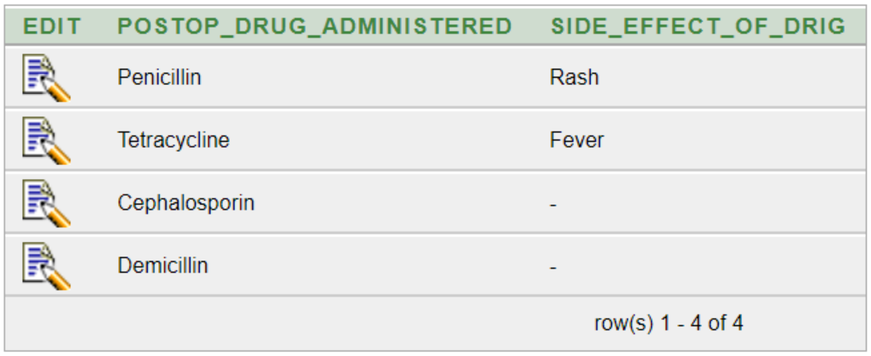
```
CREATE TABLE  "DRUGS"
   (       "POSTOP_DRUG_ADMINISTERED" VARCHAR2(4000),
           "SIDE_EFFECT_OF_DRIG" VARCHAR2(4000),
            CONSTRAINT "DRUGS_PK" PRIMARY KEY ("POSTOP_DRUG_ADMINISTERED") ENABLE
   )
/

CREATE OR REPLACE TRIGGER  "BI_DRUGS"
  before insert on "DRUGS"
  for each row
begin
    select "DRUGS_SEQ".nextval into :NEW.POSTOP_DRUG_ADMINISTERED from dual;
end;

/
ALTER TRIGGER  "BI_DRUGS" ENABLE
/
```

<u>Functional Dependency:</u>
(DRUG_ADMINISTERED) → (SIDE_EFFECT_OF_DRUG)



Column Name	Data Type	Nullable	Default	Primary Key
POSTOP_DRUG_ADMINISTERED	VARCHAR2(4000)	No	-	1
SIDE_EFFECT_OF_DRIG	VARCHAR2(4000)	Yes	-	-
				1 - 2

Drugs Table Structure

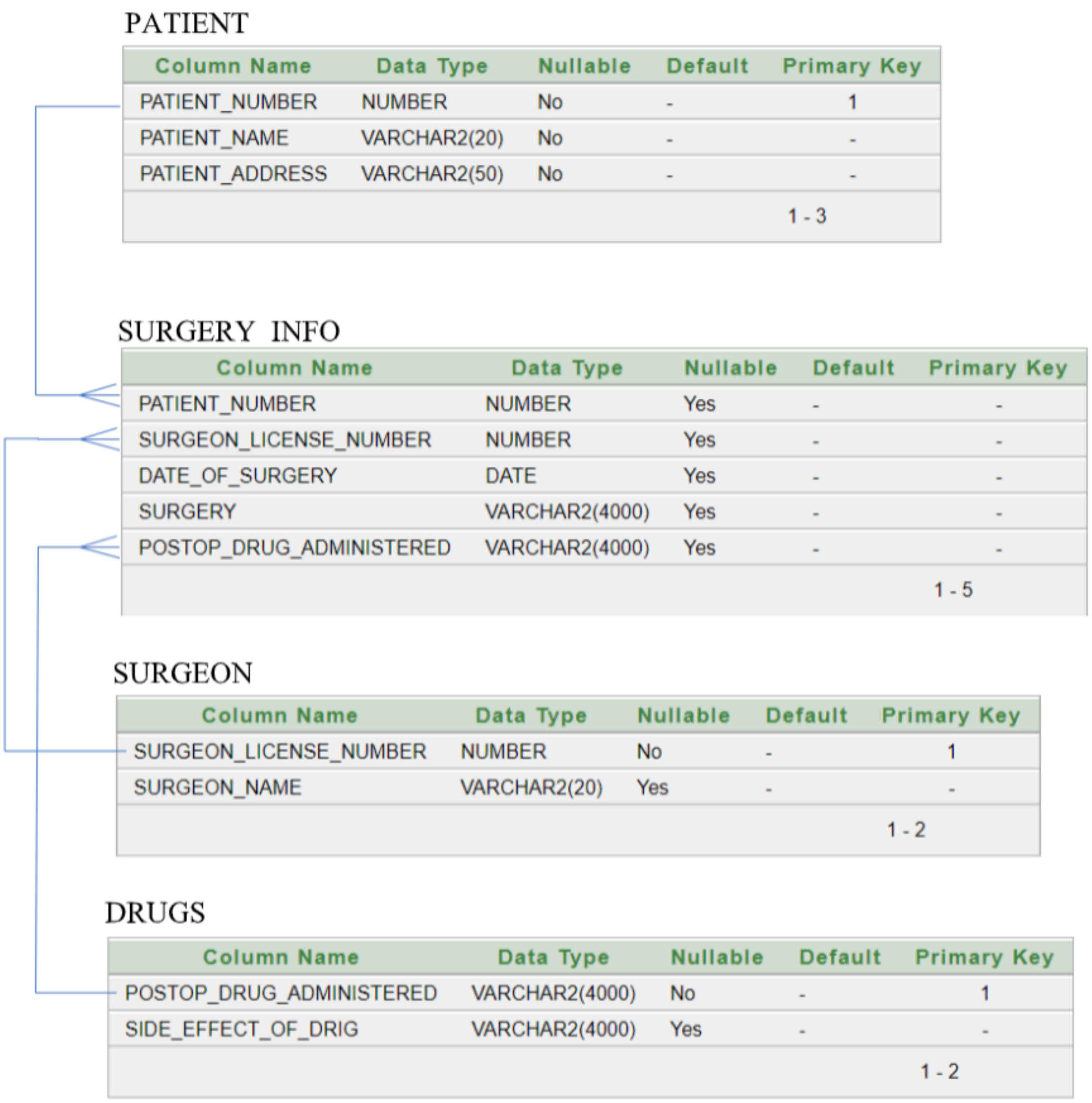


EDIT	POSTOP_DRUG_ADMINISTERED	SIDE_EFFECT_OF_DRIG
✎	Penicillin	Rash
✎	Tetracycline	Fever
✎	Cephalosporin	-
✎	Demicillin	-
		row(s) 1 - 4 of 4

Drugs Table Data

<u>Part B:</u>

<u>E/R Model:</u>



PATIENT

Column Name	Data Type	Nullable	Default	Primary Key
PATIENT_NUMBER	NUMBER	No	-	1
PATIENT_NAME	VARCHAR2(20)	No	-	-
PATIENT_ADDRESS	VARCHAR2(50)	No	-	-
				1 - 3

SURGERY INFO

Column Name	Data Type	Nullable	Default	Primary Key
PATIENT_NUMBER	NUMBER	Yes	-	-
SURGEON_LICENSE_NUMBER	NUMBER	Yes	-	-
DATE_OF_SURGERY	DATE	Yes	-	-
SURGERY	VARCHAR2(4000)	Yes	-	-
POSTOP_DRUG_ADMINISTERED	VARCHAR2(4000)	Yes	-	-
				1 - 5

SURGEON

Column Name	Data Type	Nullable	Default	Primary Key
SURGEON_LICENSE_NUMBER	NUMBER	No	-	1
SURGEON_NAME	VARCHAR2(20)	Yes	-	-
				1 - 2

DRUGS

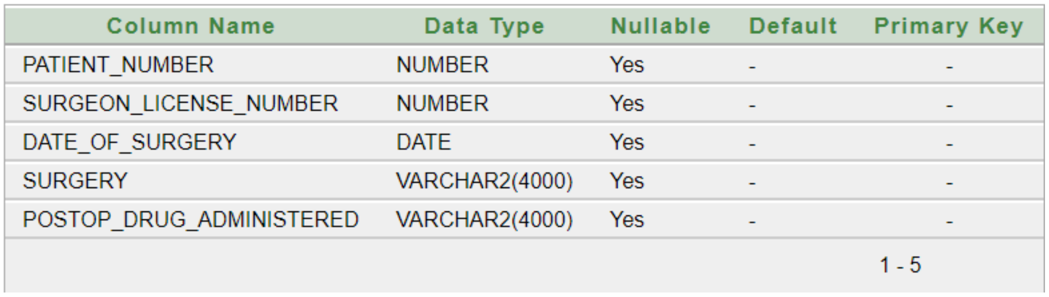
Column Name	Data Type	Nullable	Default	Primary Key
POSTOP_DRUG_ADMINISTERED	VARCHAR2(4000)	No	-	1
SIDE_EFFECT_OF_DRIG	VARCHAR2(4000)	Yes	-	-
				1 - 2

Part C:

Query 1:

A query that selects all rows from the SURGERY_INFO table and PATIENT table and performs an explicit join on the Patient_Number column

SELECT *
FROM "SURGERY_INFO" JOIN "PATIENT"
ON SURGERY_INFO.Patient_Number = PATIENT.Patient_Number;

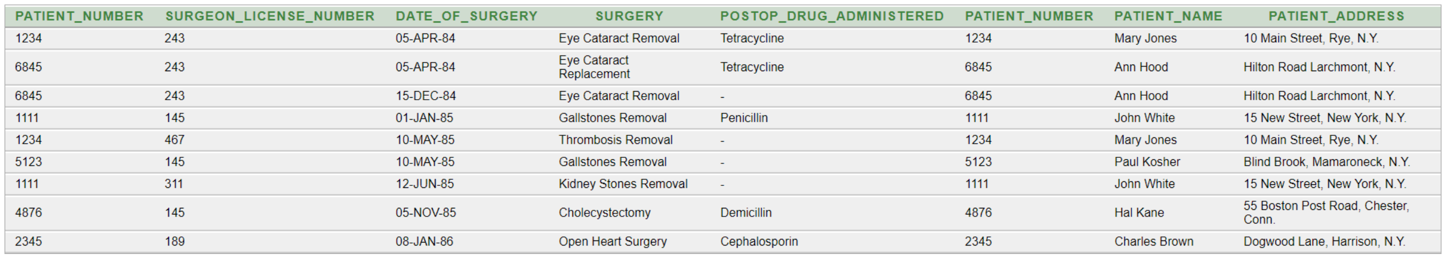


Column Name	Data Type	Nullable	Default	Primary Key
PATIENT_NUMBER	NUMBER	Yes	-	-
SURGEON_LICENSE_NUMBER	NUMBER	Yes	-	-
DATE_OF_SURGERY	DATE	Yes	-	-
SURGERY	VARCHAR2(4000)	Yes	-	-
POSTOP_DRUG_ADMINISTERED	VARCHAR2(4000)	Yes	-	-
				1 - 5

Query 2:

A query that selects all rows from the SURGERY_INFO table and PATIENT table and performs an explicit join on the Patient_Number rows and then orders the results by Date_Of_Surgery

SELECT *
FROM "SURGERY_INFO" JOIN "PATIENT"
ON SURGERY_INFO.Patient_Number = PATIENT.Patient_Number
ORDER BY Date_Of_Surgery;

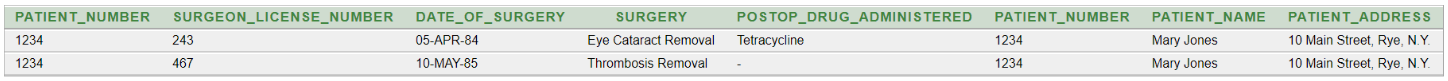


PATIENT_NUMBER	SURGEON_LICENSE_NUMBER	DATE_OF_SURGERY	SURGERY	POSTOP_DRUG_ADMINISTERED	PATIENT_NUMBER	PATIENT_NAME	PATIENT_ADDRESS
1234	243	05-APR-84	Eye Cataract Removal	Tetracycline	1234	Mary Jones	10 Main Street, Rye, N.Y.
6845	243	05-APR-84	Eye Cataract Replacement	Tetracycline	6845	Ann Hood	Hilton Road Larchmont, N.Y.
6845	243	15-DEC-84	Eye Cataract Removal	-	6845	Ann Hood	Hilton Road Larchmont, N.Y.
1111	145	01-JAN-85	Gallstones Removal	Penicillin	1111	John White	15 New Street, New York, N.Y.
1234	467	10-MAY-85	Thrombosis Removal	-	1234	Mary Jones	10 Main Street, Rye, N.Y.
5123	145	10-MAY-85	Gallstones Removal	-	5123	Paul Kosher	Blind Brook, Mamaroneck, N.Y.
1111	311	12-JUN-85	Kidney Stones Removal	-	1111	John White	15 New Street, New York, N.Y.
4876	145	05-NOV-85	Cholecystectomy	Demicillin	4876	Hal Kane	55 Boston Post Road, Chester, Conn.
2345	189	08-JAN-86	Open Heart Surgery	Cephalosporin	2345	Charles Brown	Dogwood Lane, Harrison, N.Y.

Query 3:

A query that selects all rows from the SURGERY_INFO table and PATIENT table and performs an explicit join on the Patient_Number column, and returns all rows where Pateient_Name is equal to Mary Jones

SELECT *
FROM "SURGERY_INFO" JOIN "PATIENT"
ON SURGERY_INFO.Patient_Number = PATIENT.Patient_Number
WHERE Patient_Name = 'Mary Jones';

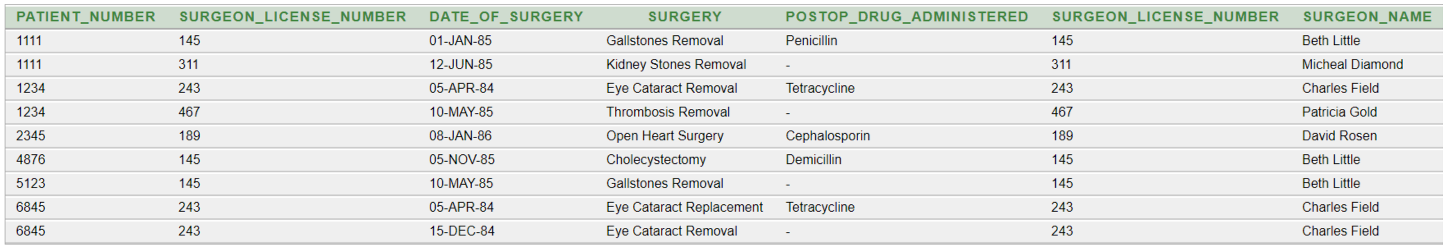


PATIENT_NUMBER	SURGEON_LICENSE_NUMBER	DATE_OF_SURGERY	SURGERY	POSTOP_DRUG_ADMINISTERED	PATIENT_NUMBER	PATIENT_NAME	PATIENT_ADDRESS
1234	243	05-APR-84	Eye Cataract Removal	Tetracycline	1234	Mary Jones	10 Main Street, Rye, N.Y.
1234	467	10-MAY-85	Thrombosis Removal	-	1234	Mary Jones	10 Main Street, Rye, N.Y.

Query 4:

A query that selects all rows from the SURGERY_INFO table and SURGEON table and performs an explicit join on the Surgeon_License_Number column

SELECT *
FROM "SURGERY_INFO" JOIN "SURGEON"
ON SURGERY_INFO.Surgeon_License_Number = SURGEON.Surgeon_License_Number;

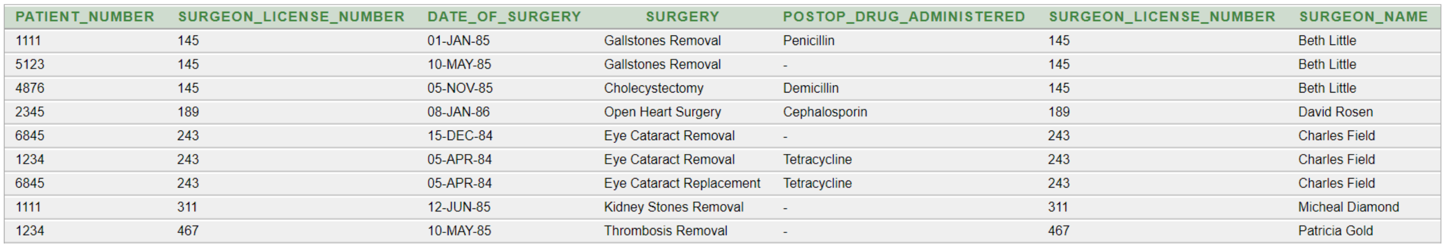


PATIENT_NUMBER	SURGEON_LICENSE_NUMBER	DATE_OF_SURGERY	SURGERY	POSTOP_DRUG_ADMINISTERED	SURGEON_LICENSE_NUMBER	SURGEON_NAME
1111	145	01-JAN-85	Gallstones Removal	Penicillin	145	Beth Little
1111	311	12-JUN-85	Kidney Stones Removal	-	311	Micheal Diamond
1234	243	05-APR-84	Eye Cataract Removal	Tetracycline	243	Charles Field
1234	467	10-MAY-85	Thrombosis Removal	-	467	Patricia Gold
2345	189	08-JAN-86	Open Heart Surgery	Cephalosporin	189	David Rosen
4876	145	05-NOV-85	Cholecystectomy	Demicillin	145	Beth Little
5123	145	10-MAY-85	Gallstones Removal	-	145	Beth Little
6845	243	05-APR-84	Eye Cataract Replacement	Tetracycline	243	Charles Field
6845	243	15-DEC-84	Eye Cataract Removal	-	243	Charles Field

Query 5:

A query that selects all rows from the SURGERY_INFO table and SURGEON table and performs an explicit join on the Surgeon_License_Number column and then orders the results by Surgeon_License_Number

SELECT *
FROM "SURGERY_INFO" JOIN "SURGEON"
ON SURGERY_INFO.Surgeon_License_Number = SURGEON.Surgeon_License_Number
ORDER BY SURGERY_INFO.Surgeon_License_Number;

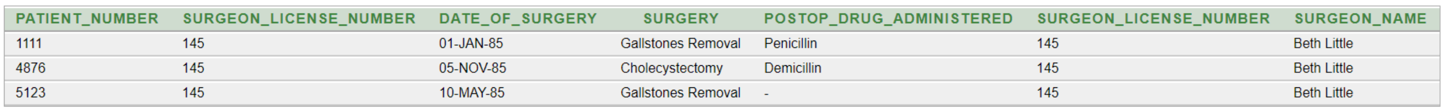


PATIENT_NUMBER	SURGEON_LICENSE_NUMBER	DATE_OF_SURGERY	SURGERY	POSTOP_DRUG_ADMINISTERED	SURGEON_LICENSE_NUMBER	SURGEON_NAME
1111	145	01-JAN-85	Gallstones Removal	Penicillin	145	Beth Little
5123	145	10-MAY-85	Gallstones Removal	-	145	Beth Little
4876	145	05-NOV-85	Cholecystectomy	Demicillin	145	Beth Little
2345	189	08-JAN-86	Open Heart Surgery	Cephalosporin	189	David Rosen
6845	243	15-DEC-84	Eye Cataract Removal	-	243	Charles Field
1234	243	05-APR-84	Eye Cataract Removal	Tetracycline	243	Charles Field
6845	243	05-APR-84	Eye Cataract Replacement	Tetracycline	243	Charles Field
1111	311	12-JUN-85	Kidney Stones Removal	-	311	Micheal Diamond
1234	467	10-MAY-85	Thrombosis Removal	-	467	Patricia Gold

Query 6:

A query that selects all rows from the SURGERY_INFO table and SURGEON table and performs an explicit join on the Surgeon_License_Number column, and returns all rows where Surgeon_Name is equal to Beth Little

SELECT *
FROM "SURGERY_INFO" JOIN "SURGEON"
ON SURGERY_INFO.Surgeon_License_Number = SURGEON.Surgeon_License_Number
WHERE Surgeon_Name = 'Beth Little';

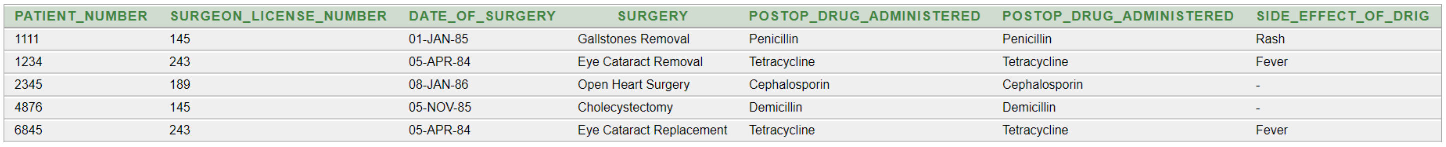


PATIENT_NUMBER	SURGEON_LICENSE_NUMBER	DATE_OF_SURGERY	SURGERY	POSTOP_DRUG_ADMINISTERED	SURGEON_LICENSE_NUMBER	SURGEON_NAME
1111	145	01-JAN-85	Gallstones Removal	Penicillin	145	Beth Little
4876	145	05-NOV-85	Cholecystectomy	Demicillin	145	Beth Little
5123	145	10-MAY-85	Gallstones Removal	-	145	Beth Little

Query 7:

A query that selects all rows from the SURGERY_INFO table and DRUGS table and performs an explicit join on the Postop_Drug_Administered column

SELECT *
FROM "SURGERY_INFO" JOIN "DRUGS"
ON SURGERY_INFO.Postop_Drug_Administered=DRUGS.Postop_Drug_Administered;

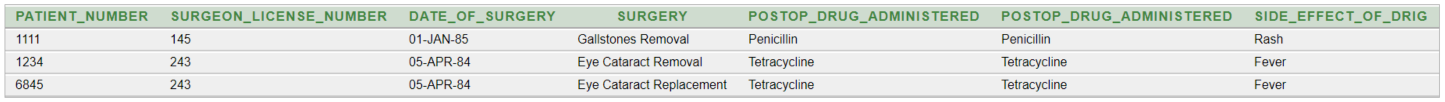


PATIENT_NUMBER	SURGEON_LICENSE_NUMBER	DATE_OF_SURGERY	SURGERY	POSTOP_DRUG_ADMINISTERED	POSTOP_DRUG_ADMINISTERED	SIDE_EFFECT_OF_DRIG
1111	145	01-JAN-85	Gallstones Removal	Penicillin	Penicillin	Rash
1234	243	05-APR-84	Eye Cataract Removal	Tetracycline	Tetracycline	Fever
2345	189	08-JAN-86	Open Heart Surgery	Cephalosporin	Cephalosporin	-
4876	145	05-NOV-85	Cholecystectomy	Demicillin	Demicillin	-
6845	243	05-APR-84	Eye Cataract Replacement	Tetracycline	Tetracycline	Fever

Query 8:

A query that selects all rows from the SURGERY_INFO table and DRUGS table and performs an explicit join on the Postop_Drug_Administered column and then displays the instances where the drug administered had a side effect

SELECT *
FROM "SURGERY_INFO" JOIN "DRUGS"
ON SURGERY_INFO.Postop_Drug_Administered=DRUGS.Postop_Drug_Administered
WHERE Side_Effect_Of_Drig IS NOT NULL;

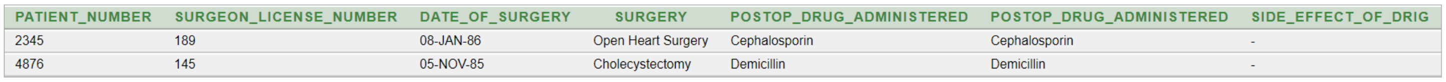


PATIENT_NUMBER	SURGEON_LICENSE_NUMBER	DATE_OF_SURGERY	SURGERY	POSTOP_DRUG_ADMINISTERED	POSTOP_DRUG_ADMINISTERED	SIDE_EFFECT_OF_DRIG
1111	145	01-JAN-85	Gallstones Removal	Penicillin	Penicillin	Rash
1234	243	05-APR-84	Eye Cataract Removal	Tetracycline	Tetracycline	Fever
6845	243	05-APR-84	Eye Cataract Replacement	Tetracycline	Tetracycline	Fever

Query 9:

A query that selects all rows from the SURGERY_INFO table and DRUGS table and performs an explicit join on the Postop_Drug_Administered column, and then displays the instances where the drug administered had no side effect

SELECT *
FROM “SURGERY_INFO” JOIN “DRUGS”
ON SURGERY_INFO.Postop_Drug_Administered = DRUGS. Postop_Drug_Administered;
WHERE Side_Effect_Of_Drig IS NULL;

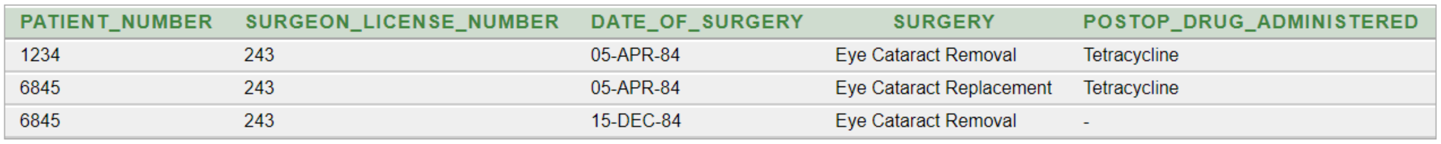


PATIENT_NUMBER	SURGEON_LICENSE_NUMBER	DATE_OF_SURGERY	SURGERY	POSTOP_DRUG_ADMINISTERED	POSTOP_DRUG_ADMINISTERED	SIDE_EFFECT_OF_DRIG
2345	189	08-JAN-86	Open Heart Surgery	Cephalosporin	Cephalosporin	-
4876	145	05-NOV-85	Cholecystectomy	Demicillin	Demicillin	-

Query 10:

A query to select all rows from SURGERY_INFO table entries where the Surgery type begins with the letters ‘Ey’

```
SELECT *
FROM "SURGERY_INFO"
WHERE Surgery LIKE 'Ey%';
```



PATIENT_NUMBER	SURGEON_LICENSE_NUMBER	DATE_OF_SURGERY	SURGERY	POSTOP_DRUG_ADMINISTERED
1234	243	05-APR-84	Eye Cataract Removal	Tetracycline
6845	243	05-APR-84	Eye Cataract Replacement	Tetracycline
6845	243	15-DEC-84	Eye Cataract Removal	-