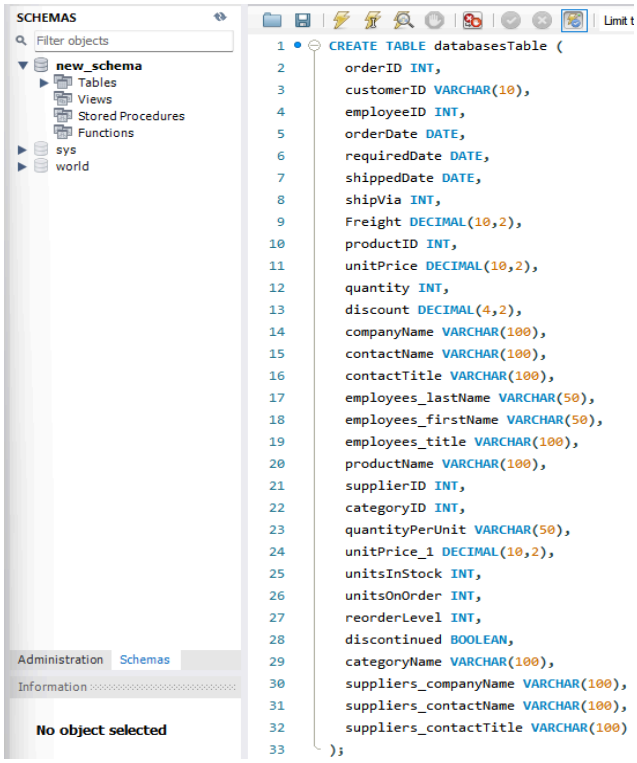
Phase 1: Database Creation & Data Loading

We decided to host the database in Docker and use MySQL Workbench.
A database and table were created in the terminal and MySQL Workbench, respectively.



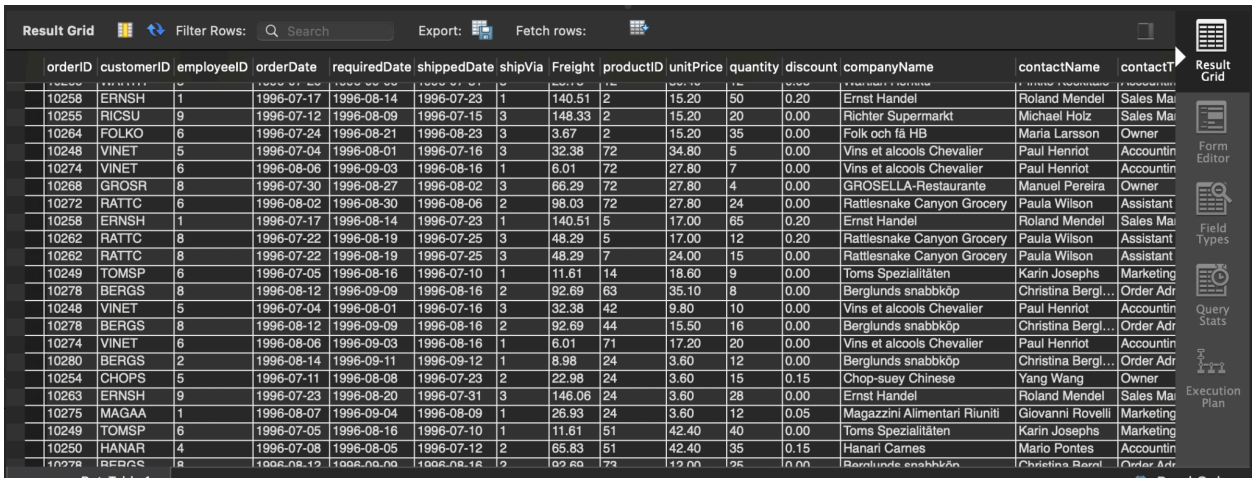
To load the TSV successfully, we had to convert the dates to use the yyyy-mm-dd format. TSV was chosen over typical CSV because it contains a field that includes a comma , which breaks the MySQL database.

orderDate	requiredDate	shippedDate
1996-7-4	1996-8-1	1996-7-16
1996-7-26	1996-9-6	1996-7-31
1996-7-17	1996-8-14	1996-7-23
1996-7-12	1996-8-9	1996-7-15
1996-7-24	1996-8-21	1996-8-23
1996-7-4	1996-8-1	1996-7-16
1996-8-6	1996-9-3	1996-8-16

From there, we were able to load the *TSV* into the table using the LOAD LOCAL INFILE command. This left us with a large table that required normalization.

```
LOAD DATA LOCAL infile '/var/lib/mysql-files/CompanyData-clean.tsv'
INTO TABLE companyDataTable
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES;
```

`SELECT * FROM companyDataTable`



Result Grid | Filter Rows: Search | Export: | Fetch rows:

orderID	customerID	employeeID	orderDate	requiredDate	shippedDate	shipVia	Freight	productID	unitPrice	quantity	discount	companyName	contactName	contactT
10258	ERNSH	1	1996-07-17	1996-08-14	1996-07-23	1	140.51	2	15.20	50	0.20	Ernst Handel	Roland Mendel	Sales Ma
10255	RICSU	9	1996-07-12	1996-08-09	1996-07-15	3	148.33	2	15.20	20	0.00	Richter Supermarkt	Michael Holz	Sales Ma
10264	FOLKO	6	1996-07-24	1996-08-21	1996-08-23	3	3.67	2	15.20	35	0.00	Folk och fä HB	Maria Larsson	Owner
10248	VINET	5	1996-07-04	1996-08-01	1996-07-16	3	32.38	72	34.80	5	0.00	Vins et alcools Chevalier	Paul Henriot	Accountin
10274	VINET	6	1996-08-06	1996-09-03	1996-08-16	1	6.01	72	27.80	7	0.00	Vins et alcools Chevalier	Paul Henriot	Accountin
10268	GROSR	8	1996-07-30	1996-08-27	1996-08-02	3	66.29	72	27.80	4	0.00	GROSELLA-Restaurante	Manuel Pereira	Owner
10272	RATTC	6	1996-08-02	1996-08-30	1996-08-06	2	98.03	72	27.80	24	0.00	Rattlesnake Canyon Grocery	Paula Wilson	Assistant
10258	ERNSH	1	1996-07-17	1996-08-14	1996-07-23	1	140.51	5	17.00	65	0.20	Ernst Handel	Roland Mendel	Sales Ma
10262	RATTC	8	1996-07-22	1996-08-19	1996-07-25	3	48.29	5	17.00	12	0.20	Rattlesnake Canyon Grocery	Paula Wilson	Assistant
10262	RATTC	8	1996-07-22	1996-08-19	1996-07-25	3	48.29	7	24.00	15	0.00	Rattlesnake Canyon Grocery	Paula Wilson	Assistant
10249	TOMSP	6	1996-07-05	1996-08-16	1996-07-10	1	11.61	14	18.60	9	0.00	Toms Spezialitäten	Karin Josephs	Marketing
10278	BERGS	8	1996-08-12	1996-09-09	1996-08-16	2	92.69	63	35.10	8	0.00	Berglunds snabbköp	Christina Bergl...	Order Adr
10248	VINET	5	1996-07-04	1996-08-01	1996-07-16	3	32.38	42	9.80	10	0.00	Vins et alcools Chevalier	Paul Henriot	Accountin
10278	BERGS	8	1996-08-12	1996-09-09	1996-08-16	2	92.69	44	15.50	16	0.00	Berglunds snabbköp	Christina Bergl...	Order Adr
10274	VINET	6	1996-08-06	1996-09-03	1996-08-16	1	6.01	71	17.20	20	0.00	Vins et alcools Chevalier	Paul Henriot	Accountin
10280	BERGS	2	1996-08-14	1996-09-11	1996-09-12	1	8.98	24	3.60	12	0.00	Berglunds snabbköp	Christina Bergl...	Order Adr
10254	CHOPS	5	1996-07-11	1996-08-08	1996-07-23	2	22.98	24	3.60	15	0.15	Chop-suey Chinese	Yang Wang	Owner
10263	ERNSH	9	1996-07-23	1996-08-20	1996-07-31	3	146.06	24	3.60	28	0.00	Ernst Handel	Roland Mendel	Sales Ma
10275	MAGAA	1	1996-08-07	1996-09-04	1996-08-09	1	26.93	24	3.60	12	0.05	Magazzini Alimentari Riuniti	Giovanni Rovelli	Marketing
10249	TOMSP	6	1996-07-05	1996-08-16	1996-07-10	1	11.61	51	42.40	40	0.00	Toms Spezialitäten	Karin Josephs	Marketing
10250	HANAR	4	1996-07-08	1996-08-05	1996-07-12	2	65.83	51	42.40	35	0.15	Hanari Carnes	Mario Pontes	Accountin

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Phase 2: Normalization and Denormalization

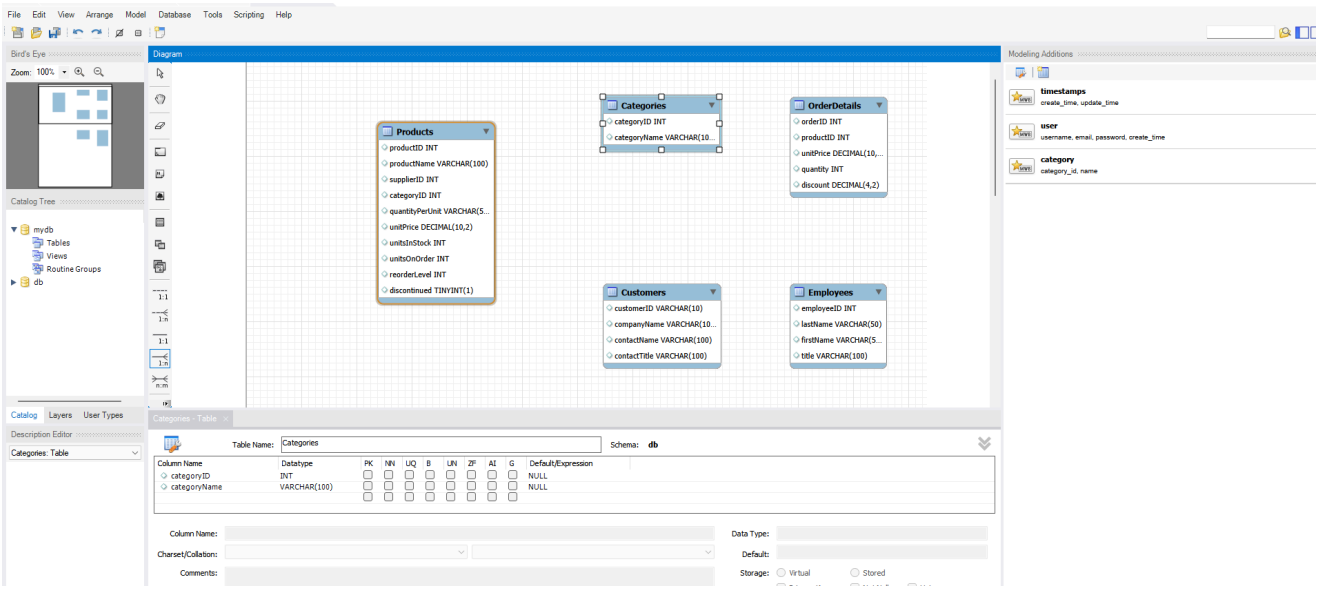
For the second phase, we normalized the `companyDataTable` into seven distinct tables: categories, customers, employees, order details, orders, products, and suppliers. Doing so makes the data significantly easier to work with, and the 3NF removes the vast majority of duplicate information.

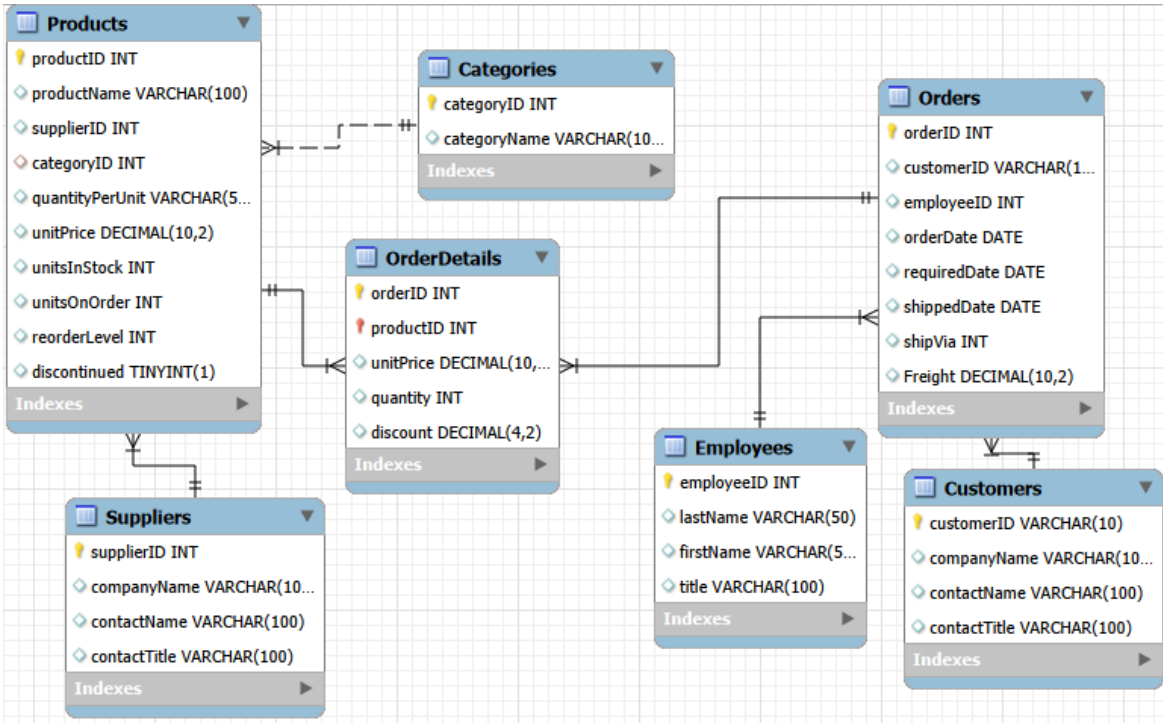
```
mysql> SHOW TABLES;
+-----------------------+
| Tables_in_db          |
+-----------------------+
| Categories            |
| Customers             |
| Employees             |
| OrderDetails          |
| Orders                |
| Products              |
| Suppliers             |
| companyDataTable      |
| quantity_revenue_view |
+-----------------------+
9 rows in set (0.002 sec)
```

Additional normalization information is in the **tables.sql** file

Phase 3: Database Diagram Design (ER)

We created the ER diagram using the built-in tool in Workbench and added foreign and primary keys, as well as relations, there as well. Due to the sequential nature of `orderId`, we decided to mark it as auto-incrementing as well.





Phase 4: Table Alterations

For Phase 4, we established the primary keys in all tables, then created the foreign keys and auto-increments.

The full SQL for phase 4 is in the **alterations.sql** file.

Phase 5: Views, Triggers, and Stored Procedures

```sql

-- create view that shows quantity sold and revenue generated by each employee
CREATE VIEW IF NOT EXISTS quantity_revenue_view AS
SELECT
    employeeID, SUM(quantity) AS quantity_sold, SUM(quantity * unitPrice * (1.0 - discount)) AS revenue_generated
FROM OrderDetails
INNER JOIN Orders ON Orders.orderID=OrderDetails.orderID
GROUP BY employeeID;

SELECT * FROM quantity_revenue_view
INTO OUTFILE '/var/lib/mysql-files/dumps/db_view.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```

We began by creating a view to check the quantity of sales and total revenue for a specific employee ID. The total revenue also considers the discount rate and adjusts it accordingly.

employeeID	quantity_so...	revenue_generat...
1	3741	87219.3300
2	2784	61642.5100
3	3421	83740.6675
4	5301	123750.4050
5	1549	34917.3950
6	1745	35649.0550
7	1881	52342.4025
8	2639	53275.0775
9	1011	16577.3950

Trigger

```sql
DELIMITER //

CREATE TRIGGER reduce_stock_after_order
AFTER INSERT ON OrderDetails
FOR EACH ROW
BEGIN
    UPDATE Products
    SET unitsInStock = unitsInStock - NEW.quantity
    WHERE productID = NEW.productID;
END//

DELIMITER ;
```

Always update units in stock, and also change the delimiter to // for the inside procedure

database-systems-final > dumps > stock_after_order.csv > data

```csv
"1","Chai","39"
"2","Chang","17"
"3","Aniseed Syrup","13"
"4","Chef Anton's Cajun Seasoning","53"
"5","Chef Anton's Gumbo Mix","0"
"6","Grandma's Boysenberry Spread","120"
"7","Uncle Bob's Organic Dried Pears","15"
"8","Northwoods Cranberry Sauce","6"
"9","Mishi Kobe Niku","29"
"10","Ikura","31"
"11","Queso Cabrales","22"
"12","Queso Manchego La Pastora","86"
"13","Konbu","24"
"14","Tofu","35"
"15","Genen Shou
"16","Pavlova","29"
"17","Alice Mutton","0"
"18","Carnarvon Tigers","42"
"19","Teatime Chocolate Biscuits","25"
"20","Sir Rodney's Marmalade","40"
"21","Sir Rodney's Scones","3"
"22","Gustaf's Knäckebröd","104"
"23","Tunnbröd","61"
"24","Guaraná Fantástica","20"
"25","NuNuCa Nuß-Nougat-Creme","76"
"26","Gumbär Gummibärchen","15"
"27","Schoggi Schokolade","49"
"28","Rössle Sauerkraut","26"
"29","Thüringer Rostbratwurst","0"
"30","Nord-Ost Matjeshering","10"
"31","Gorgonzola Telino","0"
"32","Mascarpone Fabioli","9"
"33","Geitost","112"
"34","Sasquatch Ale","111"
"35","Steeleye Stout","20"
"36","Inlagd Sill","112"
"37","Gravad lax","11"
"38","Côte de Blaye","17"
"39","Chartreuse verte","69"
"40","Boston Crab Meat","123"
"41","Jack's New England Clam Chowder","85"
"42","Singaporean Hokkien Fried Mee","26"
"43","Ipoh Coffee","17"
"44","Gula Malacca","27"
"45","Rogede sild","5"
"46","Spegesild","95"
"47","Zaanse koeken","36"
"48","Chocolade","15"
"49","Maxilaku","10"
"50","Valkoinen suklaa","65"
"51","Manjimup Dried Apples","20"
"52","Filo Mix","38"
"53","Perth Pasties","0"
"54","Tourtière","21"
"55","Pâté chinois","115"
"56","Gnocchi di nonna Alice","21"
"57","Ravioli Angelo","36"
"58","Escargots de Bourgogne","62"
"59","Raclette Courdavault","79"
"60","Camembert Pierrot","19"
"61","Sirop d'érable","113"
"62","Tarte au sucre","17"
"63","Vegie-spread","24"
"64","Wimmers gute Semmelknödel","22"
"65","Louisiana Fiery Hot Pepper Sauce","76"
"66","Louisiana Hot Spiced Okra","4"
"67","Laughing Lumberjack Lager","52"
"68","Scottish Longbreads","6"
"69","Gudbrandsdalsost","26"
"70","Outback Lager","15"
"71","Flotemysost","26"
"72","Mozzarella di Giovanni","14"
"73","Röd Kaviar","101"
"74","Longlife Tofu","4"
"75","Rhönbräu Klosterbier","125"
"76","Lakkalikööri","57"
"77","Original Frankfurter grüne Soße","32"
```

Col 3: 39

Stored Procedures

```
DELIMITER //

CREATE PROCEDURE IF NOT EXISTS GetProductsToRestock()
BEGIN
    SELECT productID, productName, unitsInStock, reorderLevel
    FROM Products
    WHERE
        unitsInStock < reorderLevel
        AND
        discontinued = 0;
END //

DELIMITER ;
```

Same as trigger, change the delimiter, and create a procedure to make a list if units in stock are lower than reorder level. Also, do not add to the list if the product is discontinued.

```
bash-5.1# mysql --local-infile=1 -uroot -p db < /var/lib/mysql-files/sql/stock_check.sql
Enter password:
productID       productName     unitsInStock    reorderLevel
2       Chang   17      25
3       Aniseed Syrup   13      25
11      Queso Cabrales  22      30
21      Sir Rodney's Scones     3       5
30      Nord-Ost Matjeshering   10      15
31      Gorgonzola Telino       0       20
32      Mascarpone Fabioli      9       25
37      Gravad lax      11      25
43      Ipoh Coffee     17      25
45      Rogede sild     5       15
48      Chocolade       15      25
49      Maxilaku        10      15
56      Gnocchi di nonna Alice  21      30
64      Wimmers gute Semmelkn?del       22      30
66      Louisiana Hot Spiced Okra       4       20
68      Scottish Longbreads     6       15
70      Outback Lager   15      30
74      Longlife Tofu   4       5
```

Phase 6: User Management and Privileges

The users and privileges were added as follows:

```
80 •  CREATE USER 'AlexSmith'@'localhost' IDENTIFIED BY 'asmith2013flikka!';
81
82    -- grant admin privileges to AlexSmith
83 •  GRANT ALL PRIVILEGES ON *.* TO 'AlexSmith'@'localhost' WITH GRANT OPTION;
84
85
86 •  CREATE USER 'JamieLee'@'localhost' IDENTIFIED BY 'jlee2021kvinna!';
87    -- grant read-only privileges to JamieLee
88 •  GRANT SELECT ON *.* TO 'JamieLee'@'localhost';
89
90 •  FLUSH PRIVILEGES;
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
9	15:17:10	CREATE TABLE OrderDetails AS SELECT ord...	1000 row(s) affected Records: 1000 Duplicates: ...	0.047 sec
10	15:50:49	CREATE USER 'AlexSmith'@'localhost' IDENTIFI...	0 row(s) affected	0.078 sec
11	15:50:49	GRANT ALL PRIVILEGES ON *.* TO 'AlexSmith'...	0 row(s) affected, 1 warning(s): 1285 MySQL is sta...	0.016 sec
12	15:50:49	CREATE USER 'JamieLee'@'localhost' IDENTIFI...	0 row(s) affected	0.015 sec
13	15:50:49	GRANT SELECT ON *.* TO 'JamieLee'@'localhost'	0 row(s) affected, 1 warning(s): 1285 MySQL is sta...	0.016 sec
14	15:50:49	FLUSH PRIVILEGES	0 row(s) affected, 1 warning(s): 1681 'FLUSH PRI...	0.015 sec

Phase 7: Database Backup

For backing up the database, we chose to use the `mysqldump` utility. This creates a full local backup that can be stored on any external hard drive. You can create this using the `dump.sh` script.

Contribution table

Portion	E Gunderson	Y Cho	Y Ensley
Phase 1	10	0	90
Phase 2	0	0	100
Phase 3	100	0	0
Phase 4	0	0	100
Phase 5	0	100	0
Phase 6	100	0	0

Phase 7	0	0	100
Report	20	80	0
Documentation	10	0	90