COIS 3400 Database Project

Multiple Warehouse Inventory Management System Database

Ryland Whillans

Database Overview

This database will be the backend for a multiple-warehouse inventory management system similar to systems developed by Magento[1] and Brightpearl[2]. It will keep track of products, warehouses, customers, and suppliers; as well as recording customer orders and shipments from suppliers.

The database will keep track of each location a product is stored at within a warehouse using a system based on the one used by Brightpearl, where location is stored as a string separated into parts representing tiers of location (such as aisle.bay.shelf.bin).

[1] https://marketplace.magento.com/amasty-module-multi-warehouse-inventory.html

[2] https://www.brightpearl.com/multiple-warehouse-inventory-system/

Database Design

Database Schema:

product(product_id, product_name, description, unit_price, height, width, length, weight)

warehouse(warehouse_id, address, phone_number)

product_location(product_id, warehouse_id, storage_location, quantity)

customer(customer_id, first_name, last_name, dob, email_address, phone_number)

customer_order(order_id, customer_id, date, shipping_address)

order_details(order_id, product_id, quantity, sale_price)

supplier(supplier_id, supplier_name, address, phone_number)

supplier_shipment(shipment_id, supplier_id, warehouse_id, date)

shipment_details(shipment_id, product_id, quantity, unit_cost)

Assumptions:

Products can be stored in multiple locations in the same warehouse.

Multiple products can be stored in the same warehouse.

Multiple products different products can be stored in the same location

Multiple warehouses can have the same storage location code

Order and shipment IDs are unique in the database, not to customers/suppliers

Orders and shipments can contain multiple different products

Every customer has made at least one order

A customer does not have a fixed shipping address

Every supplier has made at least one shipment

Supplier names are not unique

Every shipment is made to only one warehouse

A warehouse may have received no shipments

Every product has been shipped by a supplier

A product can be shipped by multiple suppliers

A product may have never been ordered

A product’s cost may differ between orders

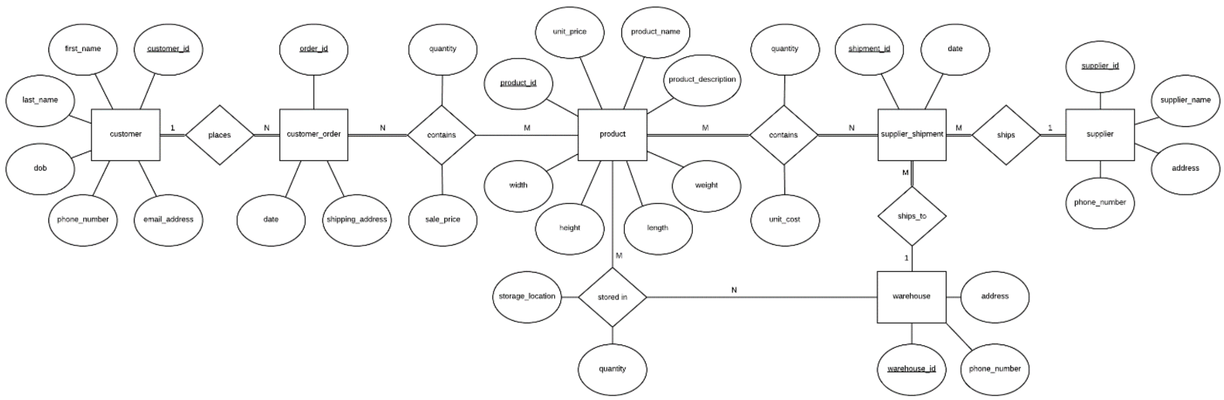
Products with the same name may have different descriptions, price, and dimensions

A product may be in 0 warehouses (out of stock)

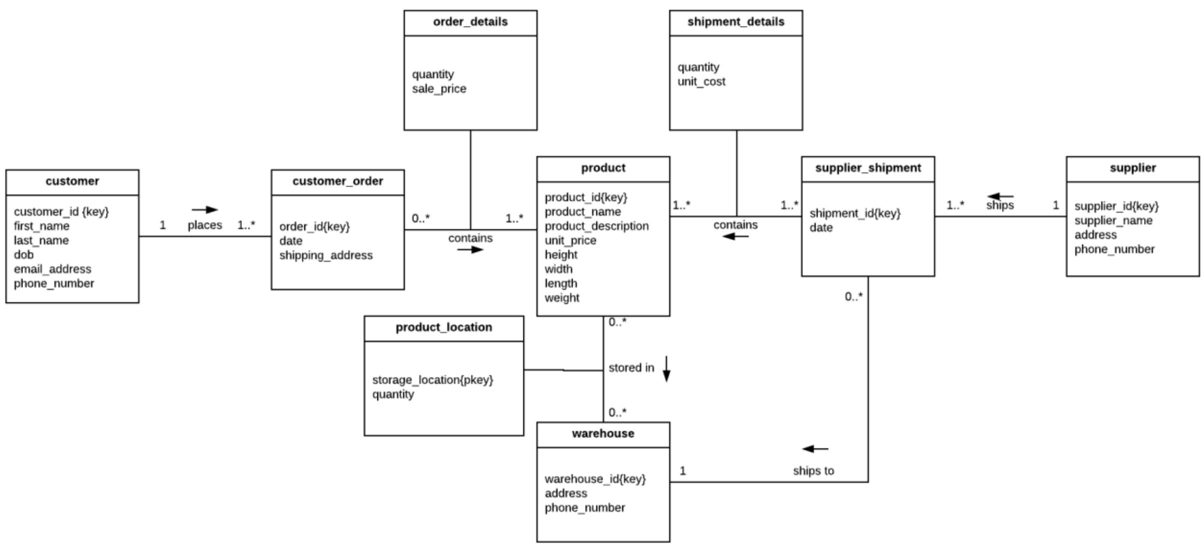
There is no dependency between any address and phone number

Addresses, email addresses, and phone numbers are not unique in any table

ER Diagram



Class Diagram



Database Analysis

<u>1NF</u>

Every table in the database has a primary key and there are no repeated or multi-valued attributes. Thus, the database is in 1NF.

<u>2NF</u>

There are 3 tables that have composite keys

- product_location: Since a product can be stored in multiple locations and multiple locations can store the same product, no subset of {product_id, warehouse_id, storage_location} can determine quantity
- order_details: Since unit_price can change, it is not possible to determine sale_price from product_id. Thus, no subset of {order_id, product_id} can determine either quantity or sale_price
- shipment_details: Since unit_cost and unit_price can change, it is not possible to determine unit_cost from product_id or shipment_id. Thus, no proper subset of {shipment_id, product_id} can determine either quantity or unit_cost

There are also no attributes of any table that are uniform, so no proper subset of any table's key can be used to determine any other attribute. Thus, the table is in 2NF.

<u>3NF</u>

product: (product_name, description, unit_price, height, width, length, weight) are independent since product names are not unique.

warehouse: (address, phone_number) are independent since address and phone number are unrelated.

product_location: (quantity) is a single attribute.

customer: (first_name, last_name, dob, email_address, phone_number) are independent.

customer_order: (customer_id, date, shipping_address) are independent since customers can use multiple shipping addresses.

order_details: (quantity, sale_price) are independent.

supplier: (supplier_name, address, phone_number) are independent since supplier names are not unique, addresses can be shared, and addresses and phone numbers are unrelated.

supplier_shipment: (supplier_id, warehouse_id, date) are independent.

shipment_details: (quantity, unit_cost) are independent.

All non-key attributes are inpedendant. Thus, the database is in 3NF

<u>BCNF</u>

Each table has only a single candidate key. Thus, the database is in BCNF

Database Implementation

```sql
CREATE DATABASE inventory_management;
USE inventory_management;
CREATE TABLE customer(
    customer_id INT,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50),
    dob DATE,
    email_address VARCHAR(50) NOT NULL,
    phone_number VARCHAR(15),
    PRIMARY KEY(customer_id));
CREATE TABLE product(
    product_id INT,
    product_name VARCHAR(50) NOT NULL,
    description VARCHAR(1000),
    unit_price DOUBLE(15,2) NOT NULL,
    height FLOAT,
    width FLOAT,
    length FLOAT,
    weight FLOAT,
    PRIMARY KEY(product_id));
CREATE TABLE supplier(
    supplier_id INT,
    supplier_name VARCHAR(50) NOT NULL,
    address VARCHAR(50),
    email_address VARCHAR(50),
    phone_number VarChar(15),
    PRIMARY KEY(supplier_id));
CREATE TABLE warehouse(
    warehouse_id INT,
    address VARCHAR(50) NOT NULL,
    phone_number VARCHAR(15),
    PRIMARY KEY(warehouse_id));
CREATE TABLE customer_order(
    order_id INT,
    customer_id INT NOT NULL,
    order_date DATE NOT NULL,
    shipping_address VARCHAR(50) NOT NULL,
    PRIMARY KEY(order_id),
    FOREIGN KEY(customer_id) REFERENCES customer(customer_id));
CREATE TABLE order_details(
    order_id INT,
    product_id INT,
    quantity INT NOT NULL,
    sale_price DOUBLE(15,2) NOT NULL,
    PRIMARY KEY(order_id, product_id),
    FOREIGN KEY(order_id) REFERENCES customer_order(order_id),
    FOREIGN KEY(product_id) REFERENCES product(product_id));
CREATE TABLE supplier_shipment(
    shipment_id INT,
    supplier_id INT NOT NULL,
    warehouse_id INT NOT NULL,
    shipment_date DATE NOT NULL,
    PRIMARY KEY(shipment_id),
    FOREIGN KEY(supplier_id) REFERENCES supplier(supplier_id),
    FOREIGN KEY(warehouse_id) REFERENCES warehouse(warehouse_id));
CREATE TABLE shipment_details(
    shipment_id INT,
    product_id INT,
    quantity INT NOT NULL,
    unit_cost DOUBLE(15,2) NOT NULL,
    PRIMARY KEY(shipment_id, product_id),
    FOREIGN KEY(shipment_id) REFERENCES supplier_shipment(shipment_id),
    FOREIGN KEY(product_id) REFERENCES product(product_id));
```

```sql
CREATE TABLE product_location(
    product_id INT,
    warehouse_id INT,
    storage_location VARCHAR(20),
    quantity INT NOT NULL,
    PRIMARY KEY(product_id, warehouse_id, storage_location),
    FOREIGN KEY(product_id) REFERENCES product(product_id),
    FOREIGN KEY(warehouse_id) REFERENCES warehouse(warehouse_id));
```

✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0059 seconds.)

```
CREATE DATABASE inventory_management
```

[Edit inline] [Edit] [Create PHP code]

✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0005 seconds.)

```
USE inventory_management
```

✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0422 seconds.)

```
CREATE TABLE customer( customer_id INT, first_name VARCHAR(50) NOT NULL, last_name VARCHAR(50), dob DATE, email_address VARCHAR(50) NOT NULL, phone_number VARCHAR(15), PRIMARY KEY(customer_id))
```

✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0291 seconds.)

```
CREATE TABLE product( product_id INT, product_name VARCHAR(50) NOT NULL, description VARCHAR(1000), unit_price DOUBLE(15,2) NOT NULL, height FLOAT, width FLOAT, length FLOAT, weight FLOAT, PRIMARY KEY(product_id))
```

✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0260 seconds.)

```
CREATE TABLE supplier( supplier_id INT, supplier_name VARCHAR(50) NOT NULL, address VARCHAR(50), email_address VARCHAR(50), phone_number VarChar(15), PRIMARY KEY(supplier_id))
```

✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0223 seconds.)

```
CREATE TABLE warehouse( warehouse_id INT, address VARCHAR(50) NOT NULL, phone_number VARCHAR(15), PRIMARY KEY(warehouse_id))
```

✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0303 seconds.)

```
CREATE TABLE customer_order( order_id INT, customer_id INT NOT NULL, order_date DATE NOT NULL, shipping_address VARCHAR(50) NOT NULL, PRIMARY KEY(order_id), FOREIGN KEY(customer_id) REFERENCES customer(customer_id))
```

✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0291 seconds.)

```
CREATE TABLE order_details( order_id INT, product_id INT, quantity INT NOT NULL, sale_price DOUBLE(15,2) NOT NULL, PRIMARY KEY(order_id, product_id), FOREIGN KEY(order_id) REFERENCES customer_order(order_id), FOREIGN KEY(product_id) REFERENCES product(product_id))
```

✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0358 seconds.)

```
CREATE TABLE supplier_shipment( shipment_id INT, supplier_id INT NOT NULL, warehouse_id INT NOT NULL, shipment_date DATE NOT NULL, PRIMARY KEY(shipment_id), FOREIGN KEY(supplier_id) REFERENCES supplier(supplier_id), FOREIGN KEY(warehouse_id) REFERENCES warehouse(warehouse_id))
```

✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0248 seconds.)

```
CREATE TABLE shipment_details( shipment_id INT, product_id INT, quantity INT NOT NULL, unit_cost DOUBLE(15,2) NOT NULL, PRIMARY KEY(shipment_id, product_id), FOREIGN KEY(shipment_id) REFERENCES supplier_shipment(shipment_id), FOREIGN KEY(product_id) REFERENCES product(product_id))
```

✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0310 seconds.)

```
CREATE TABLE product_location( product_id INT, warehouse_id INT, storage_location VARCHAR(20), quantity INT NOT NULL, PRIMARY KEY(product_id, warehouse_id, storage_location), FOREIGN KEY(product_id) REFERENCES product(product_id), FOREIGN KEY(warehouse_id) REFERENCES warehouse(warehouse_id))
```

Auto-increment added to customer.customer_id, product.product_id, supplier,supplier_id, warehouse.warehouse_id, customer_order.order_id, and supplier_shipment.shipment_id after initial implementation. It would have been to much work to recreate and repopulate all tables so it is not included here.

Data Population

customer

Showing rows 0 - 24 (100 total, Query took 0.0027 seconds.)

`SELECT * FROM `customer``

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

+ Options

	customer_id	first_name	last_name	dob	email_address	phone_number
Edit Copy Delete	1	Irvin	Lammenga	1992-03-14	ilammenga0@cpanel.net	1141108747
Edit Copy Delete	2	Angy	MacKellar	2009-11-21	amackellar1@ocn.ne.jp	5715147000
Edit Copy Delete	3	Arabel	Ryce	1988-08-24	aryce2@yolasite.com	9771490437
Edit Copy Delete	4	Rosaline	Heck	2001-04-16	rheck3@jimdo.com	6811934012
Edit Copy Delete	5	Quintin	Toffanelli	1960-05-04	qtoffanelli4@washington.edu	5404227322
Edit Copy Delete	6	Bunnie	Beckett	1943-09-20	bbeckett5@simplemachines.org	4467165759
Edit Copy Delete	7	Gwenore	Sawnwy	1973-03-16	gsawnwy6@bigcartel.com	4565094243
Edit Copy Delete	8	Duane	Klimentyonok	1991-10-15	dklimentyonok7@infoseek.co.jp	4981725315
Edit Copy Delete	9	Matthew	Lemm	1934-06-08	mlemm8@tuttocitta.it	7189909875
Edit Copy Delete	10	Alvera	todor	1943-07-12	atodor9@nationalgeographic.com	2885781683
Edit Copy Delete	11	Ashia	Tunniclisse	1988-05-28	atunniclissea@amazon.co.jp	5643350824
Edit Copy Delete	12	Arlana	Ubsdale	1982-06-15	aubsdaleb@ning.com	3828607042
Edit Copy Delete	13	Kaylil	Murrigans	1929-12-30	kmurrigansc@chron.com	3962361457
Edit Copy Delete	14	Shelden	Mil	1963-09-01	smild@digg.com	9556999092
Edit Copy Delete	15	Petronella	Ludlamme	1977-09-11	pludlammee@squidoo.com	3034331136
Edit Copy Delete	16	Reena	Segebrecht	1930-03-16	rsegebrechtf@smh.com.au	3602211978
Edit Copy Delete	17	Maurene	Bonhome	1970-12-23	mbonhomeg@hao123.com	1146046374
Edit Copy Delete	18	Zilvia	Traske	1996-07-26	ztraskeh@comsenz.com	4358411921
Edit Copy Delete	19	Paulo	Sculpher	1967-09-25	psculpheri@npr.org	8711541287
Edit Copy Delete	20	Payton	Cottham	1977-09-15	pcotthamj@omniture.com	7261363574
Edit Copy Delete	21	Rolf	Bilbee	1979-04-13	rbilbeek@stanford.edu	6805209292
Edit Copy Delete	22	Jens	Carlton	1981-02-12	jcarltonl@seesaa.net	3074593705
Edit Copy Delete	23	Brockie	Bratchell	1987-03-30	bbratchellm@ucsd.edu	1086808144
Edit Copy Delete	24	Gerda	Foulgham	1990-11-30	gfoulghamn@hao123.com	8219529804
Edit Copy Delete	25	Gerome	Osgordby	1984-10-04	gosgordbyo@t.co	1406815127

customer_order

Showing rows 0 - 499 (1000 total, Query took 0.0033 seconds.)

`SELECT * FROM `customer_order``

1 > >> | Number of rows: 500 Filter rows: Search this table Sort by key:

+ Options

	order_id	customer_id	order_date	shipping_address
Edit Copy Delete	1	7	2008-06-10	8927 Trailsway Crossing
Edit Copy Delete	2	18	2018-08-10	390 Lotheville Road
Edit Copy Delete	3	84	2015-02-08	82540 Iowa Trail
Edit Copy Delete	4	26	2006-11-18	27974 Kropf Trail
Edit Copy Delete	5	35	2009-10-09	0 Sachtjen Street
Edit Copy Delete	6	23	2001-07-29	687 Wayridge Parkway
Edit Copy Delete	7	30	2010-04-21	39 Meadow Valley Park
Edit Copy Delete	8	38	2013-01-13	11582 Anhalt Point
Edit Copy Delete	9	98	2013-09-20	8 Golf Court
Edit Copy Delete	10	41	2004-09-28	580 Doe Crossing Circle
Edit Copy Delete	11	61	2015-10-10	45 Grover Trail
Edit Copy Delete	12	57	2005-05-15	521 Division Point
Edit Copy Delete	13	2	2015-04-15	837 Corben Drive
Edit Copy Delete	14	94	2011-02-07	166 Schmedeman Park
Edit Copy Delete	15	31	2013-09-11	589 Bartillon Hill
Edit Copy Delete	16	50	2004-11-02	08 Manitowish Trail
Edit Copy Delete	17	90	2000-06-07	340 Iowa Lane
Edit Copy Delete	18	10	2003-10-22	1 Lillian Center
Edit Copy Delete	19	4	2003-08-25	8833 Parkside Center
Edit Copy Delete	20	95	2018-07-03	902 Arrowood Hill
Edit Copy Delete	21	46	2009-12-15	1437 Manley Street
Edit Copy Delete	22	56	2011-07-11	34 Milwaukee Circle
Edit Copy Delete	23	16	2015-06-14	6 Wayridge Way
Edit Copy Delete	24	64	2001-09-05	02969 Sugar Avenue
Edit Copy Delete	25	54	2018-08-25	31926 Lakeland Road
Edit Copy Delete	26	20	2014-02-21	69408 Stone Corner Place
Edit Copy Delete	27	73	2014-07-17	4 Graedel Avenue
Console Copy Delete	28	82	2016-03-31	6205 Independence Place

order_details

Showing rows 0 - 24 (8834 total, Query took 0.0031 seconds.)

`SELECT * FROM `order_details``

1 > >> | Number of rows: 25 Filter rows: Search this table

+ Options

	order_id	product_id	quantity	sale_price
Edit Copy Delete	1	12	24	571.65
Edit Copy Delete	1	16	95	502.04
Edit Copy Delete	1	21	69	149.79
Edit Copy Delete	1	40	11	921.22
Edit Copy Delete	1	73	71	922.67
Edit Copy Delete	1	142	8	629.72
Edit Copy Delete	1	145	32	809.69
Edit Copy Delete	1	170	98	764.09
Edit Copy Delete	1	185	46	81.05
Edit Copy Delete	1	188	94	952.06
Edit Copy Delete	1	209	76	38.09
Edit Copy Delete	1	217	96	938.23
Edit Copy Delete	1	245	97	294.94
Edit Copy Delete	2	19	99	592.77
Edit Copy Delete	2	120	83	115.41
Edit Copy Delete	2	184	65	249.84
Edit Copy Delete	2	232	51	805.45
Edit Copy Delete	3	3	35	748.14
Edit Copy Delete	3	25	27	499.13
Edit Copy Delete	3	31	8	864.21
Edit Copy Delete	3	180	37	312.07
Edit Copy Delete	3	248	93	237.72
Edit Copy Delete	4	14	4	591.90
Edit Copy Delete	4	34	82	914.11
Edit Copy Delete	4	78	7	650.95

product

Showing rows 0 - 24 (250 total, Query took 0.0030 seconds.)

`SELECT * FROM `product``

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

+ Options

	product_id	product_name	description	unit_price	height	width	length	weight
Edit Copy Delete	1	ipsum dolor	Duis ac nibh. Fusce lacus purus, aliquet at, feugi...	615.84	79.52	55.2	78.66	7.89
Edit Copy Delete	2	vestibulum ac est lacinia nisi	Donec posuere metus vitae ipsum.	306.67	68.32	49.92	3.74	5.22
Edit Copy Delete	3	dolor sit amet consectetuer adipiscing	Phasellus in felis. Donec semper sapien a libero. ...	95.21	72.05	81.8	31.35	95.39
Edit Copy Delete	4	velit id pretium iaculis diam	Nulla tellus. In sagittis dui vel nisl. Duis ac ni...	671.19	69.26	33.21	42.19	60.99
Edit Copy Delete	5	tempor turpis nec euismod	Integer aliquet, massa id lobortis convallis, tort...	103.73	27.26	34.93	98.69	62.52
Edit Copy Delete	6	at	Praesent blandit lacinia erat. Vestibulum sed magn...	378.85	66.84	34.89	33.33	17.23
Edit Copy Delete	7	turpis	Suspendisse ornare consequat lectus. In est risus,...	443.65	57.47	67.49	99.87	51.32
Edit Copy Delete	8	pellentesque eget nunc donec	Phasellus id sapien in sapien iaculis congue. Viva...	409.26	70.39	53.75	83.56	96.88
Edit Copy Delete	9	suscipit	Quisque id justo sit amet sapien dignissim vestibu...	294.49	66.04	37.71	90.53	61.01
Edit Copy Delete	10	iaculis	Quisque id justo sit amet sapien dignissim vestibu...	48.35	65.73	47.26	3.93	73.63
Edit Copy Delete	11	nisl venenatis lacinia aenean	Lorem ipsum dolor sit amet, consectetuer adipiscin...	46.41	19.27	35.91	46.94	19.67
Edit Copy Delete	12	habitasse platea dictumst maecenas	In sagittis dui vel nisl. Duis ac nibh. Fusce lacu...	255.05	10.77	18.53	52.68	93.2
Edit Copy Delete	13	ut at	Nulla tempus. Vivamus in felis eu sapien cursus ve...	707.54	12.98	69.83	57.59	32.26
Edit Copy Delete	14	id nulla ultrices	Praesent id massa id nisl venenatis lacinia. Aenea...	816.00	68.07	97.35	40.59	98.78
Edit Copy Delete	15	ultrices aliquet maecenas	Vestibulum ante ipsum primis in faucibus orci luct...	668.89	40.45	69.09	83.92	83.56
Edit Copy Delete	16	sem duis	Duis bibendum. Morbi non quam nec dui luctus rutru...	500.61	85.55	0.72	26.41	17.31
Edit Copy Delete	17	orci luctus et ultrices posuere	Donec ut mauris eget massa tempor convallis. Nulla...	517.74	25.9	34.86	92.98	27.1
Edit Copy Delete	18	non velit donec diam	Donec odio justo, sollicitudin ut, suscipit a, feu...	304.79	14.05	74.07	6.11	13.24
Edit Copy Delete	19	integer ac	Duis aliquam convallis nunc. Proin at turpis a ped...	952.68	85.67	97.35	55.5	90.2
Edit Copy Delete	20	at turpis donec	Vestibulum ante ipsum primis in faucibus orci luct...	150.00	15.23	57.39	68.54	52.54
Edit Copy Delete	21	non	Etiam justo. Etiam pretium iaculis justo. In hac h...	142.58	1.09	28.17	30.73	29.05
Edit Copy Delete	22	penatibus et magnis	Cum sociis natoque penatibus et magnis dis parturi...	589.63	91.18	39.2	40.23	31.6
Edit Copy Delete	23	nisl ut volutpat	Suspendisse accumsan tortor quis turpis. Sed ante....	447.29	48.56	33.64	11.18	46.51
Edit Copy Delete	24	libero	In est risus, auctor sed, tristique in, tempus sit...	419.57	3.02	2.97	41.4	74.3
Edit Copy Delete	25	elit ac nulla sed	Maecenas ut massa quis augue luctus tincidunt. Nul...	67.36	42.69	51.21	79.19	57.17

product_location

Showing rows 0 - 24 (2000 total, Query took 0.0071 seconds.) [quantity: 2... - 123...]

```
SELECT * FROM `product_location` ORDER BY `quantity` ASC
```

1 ▼ > >> | Number of rows: 25 ▼ Filter rows: Search this table Sort by key: None

+ Options

	product_id	warehouse_id	storage_location	quantity
Edit Copy Delete	42	9	Xalm.qF9g.nO8C.adzC	2
Edit Copy Delete	376	2	NIAM.YJM8.RDK1.u4At	8
Edit Copy Delete	231	5	JgzV.0HYg.j0uL.2Z0E	10
Edit Copy Delete	449	8	ww2F.T3So.XPHz.RUK2	20
Edit Copy Delete	139	6	t6bP.BBWD.Fe67.aJt6	25
Edit Copy Delete	231	5	Ovnf.emk8.Q7eT.ahMF	36
Edit Copy Delete	413	6	c1jk.TW49.mCa1.YePQ	36
Edit Copy Delete	178	4	NJvk.0KV0.8Icy.Ag3G	36
Edit Copy Delete	250	3	CqOk.FNYo.5SSA.2CcO	37
Edit Copy Delete	346	10	zy1W.SFy5.aTEs.a2yd	45
Edit Copy Delete	410	5	R4Nv.046q.uqKa.OkMm	46
Edit Copy Delete	251	5	0oNi.2ffv.vDDm.EVII	49
Edit Copy Delete	387	4	XgKU.q50P.8qxg.8Kme	49
Edit Copy Delete	37	1	jY2I.PJBE.0RS6.kZ25	72
Edit Copy Delete	410	10	VpOK.SdMI.WwNQ.af3h	73
Edit Copy Delete	409	2	TuQ5.s6va.UtSn.ePv8	90
Edit Copy Delete	72	10	WKCu.RYD1.WEf7.BKHR	92
Edit Copy Delete	463	10	g5jb.M0N1.ZJNj.vMVZ	99
Edit Copy Delete	264	9	DHiF.v8G0.TZFi.Whii	107
Edit Copy Delete	485	2	EUSa.k1Cd.Cyos.8yPL	108
Edit Copy Delete	468	4	HSSw.9caS.9MPf.cWPa	112
Edit Copy Delete	201	3	aw3g.6jEq.IZA9.j2we	115
Edit Copy Delete	10	7	H7j0.veVG.sr4m.3h80	115
Edit Copy Delete	211	6	wSII.kQYH.S9DI.Qm4M	119
Edit Copy Delete	336	2	q0FN.CjHv.muGo.WD6b	123

warehouse

Showing rows 0 - 9 (10 total, Query took 0.0028 seconds.)

```
SELECT * FROM `warehouse`
```

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

+ Options

	warehouse_id	address	phone_number
Edit Copy Delete	1	9161 Morning Court	9284558048
Edit Copy Delete	2	0 Gulseth Court	5831318333
Edit Copy Delete	3	88058 Kenwood Junction	5391463606
Edit Copy Delete	4	3677 Prairie Rose Hill	4666405891
Edit Copy Delete	5	698 Anthes Alley	9854634281
Edit Copy Delete	6	561 Elka Center	6619562348
Edit Copy Delete	7	31 1st Center	1346183673
Edit Copy Delete	8	3 Old Shore Point	2988262859
Edit Copy Delete	9	7542 Grover Way	4609455095
Edit Copy Delete	10	851 Fremont Center	4029530311

supplier_shipment

Showing rows 0 - 24 (500 total, Query took 0.0035 seconds.)

```
SELECT * FROM `supplier_shipment`
```

1 ▼ > >> | Show all | Number of rows: 25 ▼ Filter rows: Search this tabl

+ Options

	shipment_id	supplier_id	warehouse_id	shipment_date
Edit Copy Delete	1	17	6	2018-10-12
Edit Copy Delete	2	18	3	2018-02-07
Edit Copy Delete	3	15	8	2018-02-08
Edit Copy Delete	4	17	7	2018-06-12
Edit Copy Delete	5	4	7	2018-07-14
Edit Copy Delete	6	5	2	2018-03-25
Edit Copy Delete	7	3	4	2018-08-16
Edit Copy Delete	8	12	7	2018-06-29
Edit Copy Delete	9	20	6	2018-06-07
Edit Copy Delete	10	12	7	2018-09-29
Edit Copy Delete	11	18	3	2018-06-18
Edit Copy Delete	12	17	8	2018-11-09
Edit Copy Delete	13	14	7	2018-04-01
Edit Copy Delete	14	14	7	2018-02-15
Edit Copy Delete	15	19	10	2018-06-25
Edit Copy Delete	16	20	2	2018-06-15
Edit Copy Delete	17	19	10	2018-08-11
Edit Copy Delete	18	12	5	2018-01-27
Edit Copy Delete	19	17	10	2018-03-14
Edit Copy Delete	20	2	3	2018-01-24
Edit Copy Delete	21	20	8	2018-07-04
Edit Copy Delete	22	3	7	2018-11-23
Edit Copy Delete	23	11	2	2018-06-02
Edit Copy Delete	24	9	10	2018-06-17
Edit Copy Delete	25	11	9	2018-06-18

shipment_details

Showing rows 0 - 24 (4901 total, Query took 0.0055 seconds.)

```
SELECT * FROM `shipment_details`
```

1 ▼ > >> | Number of rows: 25 ▼ Filter rows: Search this table

+ Options

	shipment_id	product_id	quantity	unit_cost
Edit Copy Delete	1	15	8835	361.22
Edit Copy Delete	1	25	98942	549.75
Edit Copy Delete	1	37	47497	487.24
Edit Copy Delete	1	55	75316	622.86
Edit Copy Delete	1	75	89608	724.70
Edit Copy Delete	1	78	13841	131.00
Edit Copy Delete	1	95	84740	291.12
Edit Copy Delete	1	131	60976	194.93
Edit Copy Delete	1	146	18	378.64
Edit Copy Delete	1	169	8671	876.19
Edit Copy Delete	1	175	7228	949.12
Edit Copy Delete	1	194	40643	892.52
Edit Copy Delete	1	225	57525	778.32
Edit Copy Delete	2	64	83267	881.13
Edit Copy Delete	2	71	94845	233.76
Edit Copy Delete	2	75	2981	84.40
Edit Copy Delete	2	85	84954	795.26
Edit Copy Delete	2	95	89306	733.95
Edit Copy Delete	2	149	37436	267.33
Edit Copy Delete	3	20	21443	249.70
Edit Copy Delete	3	69	49077	985.25
Edit Copy Delete	3	76	81838	614.71
Edit Copy Delete	3	86	49380	529.53
Edit Copy Delete	3	158	15655	55.63
Edit Copy Delete	3	172	57215	708.74

supplier

✔ Showing rows 0 - 19 (20 total, Query took 0.0019 seconds.)

SELECT * FROM `supplier`

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

+ Options

supplier_id	supplier_name	address	email_address	phone_number
1	Kulas, Nienow and Hegmann	15 Northview Trail	jpretsell0@chronoengine.com	5296773853
2	King, Toy and Hickle	682 Veith Junction	emcphater1@china.com.cn	9622203580
3	O'Conner and Sons	0840 Lighthouse Bay Junction	tchipchase2@discuz.net	5594696788
4	Schneider, Torp and D'Amore	3 Briar Crest Lane	gmcconachie3@ezinearticles.com	9123088832
5	Little-Auer	77 Surrey Park	ncleveland4@mail.ru	3526038698
6	O'Connell Inc	8429 Debra Circle	gblampied5@parallels.com	3954029151
7	Brakus-Kertzmann	57733 Browning Court	rellerbeck6@slashdot.org	4772719207
8	Ruecker, Schulist and West	7 Reindahl Junction	apittam7@howstuffworks.com	8563104336
9	Bode-Huels	41381 Derek Parkway	ewolsey8@bbc.co.uk	6195441777
10	Senger LLC	4819 Cambridge Parkway	rmelross9@baidu.com	1273938613
11	Cruickshank-Blick	4 Homewood Plaza	dboschmanna@jugem.jp	4596585955
12	Brekke Group	509 Shelley Lane	dbessentb@irs.gov	3488991502
13	O'Reilly-Goldner	7 Summit Junction	tbundyc@netlog.com	5249853004
14	Kiehn-Greenfelder	99567 Glacier Hill Crossing	icordeuxd@liveinternet.ru	5915541948
15	Bradtke-Barrows	10 Oneill Road	igarmane@theatlantic.com	6754445895
16	Goldner Group	50689 Bunting Point	mairyf@ebay.co.uk	9046994120
17	Larkin-Pfannerstill	3 Cascade Center	hrichardesg@cocolog-nifty.com	9217790855
18	Dooley-Kertzmann	742 Anhalt Point	bforrestallh@typepad.com	2929441784
19	Rutherford and Sons	40 Eliot Road	rselvesteri@ocn.ne.jp	7832317012
20	Boyle, Legros and Roberts	7 Mandrake Street	nkerrj@istockphoto.com	4356145441

Sample SQL Queries

Name and contact information of 10 highest spending customers

SELECT first_name, last_name, phone_number, email_address

FROM customer

INNER JOIN (SELECT customer_order.customer_id, order_details.quantity*order_details.sale_price AS price

FROM customer_order

INNER JOIN order_details

ON customer_order.order_id = order_details.order_id) AS order_price

ON customer.customer_id = order_price.customer_id

GROUP BY customer.customer_id

ORDER BY SUM(order_price.price) DESC

LIMIT 10

✔ Showing rows 0 - 9 (10 total, Query took 0.0176 seconds.)

SELECT first_name, last_name, phone_number, email_address FROM customer INNER JOIN (SELECT customer_order.customer_id, order_details.quantity*order_details.sale_price AS price FROM customer_order INNER JOIN order_details ON customer_order.order_id = order_details.order_id) AS order_price ON customer.customer_id = order_price.customer_id GROUP BY customer.customer_id ORDER BY SUM(order_price.price) DESC LIMIT 10

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

+ Options

first_name	last_name	phone_number	email_address
Gwennie	McNae	6367583920	gmcnae1u@dell.com
Iseabal	Ezele	3057079769	iezele27@wired.com
Babbette	Di Claudio	9202630002	bdiclaudioy@fastcompany.com
Marcella	Zack	2122654026	mzack2n@gov.uk
Valencia	Graver	8128428464	vgraveru@spotify.com
Alberta	Swithenby	4629396162	aswithenbyv@msn.com
Brodie	Jaggers	8109226731	bjaggers2a@icq.com
Sharla	Gelly	9681974837	sgelly1z@rambler.ru
Elita	McGirl	2408568987	emcgirl24@samsung.com
Brockie	Bratchell	1086808144	bbratchellm@ucsd.edu

Name, description, and average profit per unit (avg price – avg cost) for 50 most purchased items (by total number sold)

```
SELECT product.product_name, product.description, order_price.avg_price-shipment_cost.avg_cost AS avg_profit

FROM product

INNER JOIN (SELECT order_details.product_id, SUM(order_details.quantity) AS total_quantity,
SUM(order_details.quantity*order_details.sale_price)/SUM(order_details.quantity) AS avg_price

    FROM customer_order

    INNER JOIN order_details

    ON customer_order.order_id = order_details.order_id

    GROUP BY order_details.product_id) AS order_price

ON product.product_id = order_price.product_id

INNER JOIN (SELECT shipment_details.product_id,
SUM(shipment_details.quantity*shipment_details.unit_cost)/SUM(shipment_details.quantity) AS avg_cost

    FROM supplier_shipment

    INNER JOIN shipment_details

    ON supplier_shipment.shipment_id = shipment_details.shipment_id

    GROUP BY shipment_details.product_id) AS shipment_cost

ON product.product_id = shipment_cost.product_id

ORDER BY order_price.total_quantity DESC

LIMIT 50
```

Showing rows 0 - 49 (50 total, Query took 0.0493 seconds.)

```
SELECT product.product_name, product.description, order_price.avg_price-shipment_cost.avg_cost AS avg_profit FROM product INNER JOIN (SELECT order_details.product_id, SUM(order_details.quantity) AS total_quantity, SUM(order_details.quantity*order_details.sale_price)/SUM(order_details.quantity) AS avg_price FROM customer_order INNER JOIN order_details ON customer_order.order_id = order_details.order_id GROUP BY order_details.product_id) AS order_price ON product.product_id = order_price.product_id INNER JOIN (SELECT shipment_details.product_id, SUM(shipment_details.quantity*shipment_details.unit_cost)/SUM(shipment_details.quantity) AS avg_cost FROM supplier_shipment INNER JOIN shipment_details ON supplier_shipment.shipment_id = shipment_details.shipment_id GROUP BY shipment_details.product_id) AS shipment_cost ON product.product_id = shipment_cost.product_id ORDER BY order_price[...]
```

Profiling [Edit] [Refresh]

+ Options

product_name	description	avg_profit
dictumst maecenas ut	Nulla nisl. Nunc nisl. Duis bibendum, felis sed in...	-92.246825
tristique in tempus sit	Mauris lacinia sapien quis libero. Nullam sit amet...	23.831112
eget	Morbi a ipsum.	-0.401251
tincidunt eget tempus vel pede	Aenean lectus. Pellentesque eget nunc. Donec quis ...	68.797172
lectus	Mauris enim leo, rhoncus sed, vestibulum sit amet,...	-238.496029
sollicitudin ut	Vivamus vestibulum sagittis sapien. Cum sociis nat...	95.998432
fermentum justo nec condimentum	Cras in purus eu magna vulputate luctus. Cum socii...	-140.991461
proin	Suspendisse potenti. Cras in purus eu magna vulput...	-200.865088
nulla suspendisse potenti cras in	Nam dui.	61.883218
iaculis justo in hac habitasse	Morbi non lectus. Aliquam sit amet diam in magna b...	86.129410
luctus rutrum nulla	Donec dapibus. Duis at velit eu est congue element...	-1.687224
semper porta volutpat	Maecenas pulvinar lobortis est. Phasellus sit amet...	98.472836
ut rhoncus aliquet pulvinar sed	Nulla neque libero, convallis eget, eleifend luctu...	-77.988911
at	Nam congue, risus semper porta volutpat, quam pede...	2.815053
in est risus auctor	Morbi odio odio, elementum eu, interdum eu, tincid...	3.895264
in congue etiam justo etiam	Aenean sit amet justo. Morbi ut odio. Cras mi pede...	13.717587
tempor turpis nec	Vestibulum quam sapien, varius ut, blandit non, in...	6.861943
turpis	Suspendisse ornare consequat lectus. In est risus,...	66.982303
aliquam quis turpis	Quisque id justo sit amet sapien dignissim vestibu...	125.000986
curae	Proin eu mi. Nulla ac enim. In tempor, turpis nec ...	47.201294
vivamus tortor duis mattis egestas	Nulla ut erat id mauris vulputate elementum.	-114.168240
neque sapien	Suspendisse potenti. In eleifend quam a odio. In h...	50.016059
volutpat dui	Quisque erat eros, viverra eget, congue eget, semp...	96.663173
nibh ligula nec sem duis	Pellentesque at nulla. Suspendisse potenti. Cras i...	-27.486110
pellentesque ultrices phasellus id sapien	Nunc nisl. Duis bibendum, felis sed interdum venen...	59.820406
est risus auctor	Nullam porttitor lacus at turpis. Donec posuere me...	-328.316362
mi	Donec quis orci eget orci vehicula condimentum. Cu...	127.031041
vulputate justo	Nam dui. Proin leo odio, porttitor id, consequat i...	120.426647
sit amet consectetuer	Nulla mollis molestie lorem. Quisque ut erat. Cura...	183.655705
justo nec condimentum	Maecenas rhoncus aliquam lacus. Morbi quis tortor ...	17.448736
pellentesque eget nunc donec	Phasellus id sapien in sapien iaculis congue. Viva...	-5.721121
pellentesque	Duis bibendum, felis sed interdum venenatis, turpi...	-12.168134
nulla quisque	Praesent blandit.	30.705672
volutpat dui	Nullam orci pede, venenatis non, sodales sed, tinc...	11.749621
risus	Etiam faucibus cursus urna. Ut tellus. Nulla ut er...	16.423773
nulla	Proin eu mi. Nulla ac enim. In tempor, turpis nec ...	-5.866717
turpis integer	Donec posuere metus vitae ipsum. Aliquam non mauri...	136.531078
libero nam dui proin	In quis justo. Maecenas rhoncus aliquam lacus. Mor...	10.819627
ipsum aliquam	Proin at turpis a pede posuere nonummy. Integer no...	15.462582
sollicitudin ut	Aliquam quis turpis eget elit sodales scelerisque....	25.535618
at nulla suspendisse potenti	Aliquam non mauris. Morbi non lectus. Aliquam sit ...	92.446347
non velit donec diam	Donec odio justo, sollicitudin ut, suscipit a, feu...	22.575678
dapibus nulla suscipit ligula	Aenean sit amet justo. Morbi ut odio. Cras mi pede...	69.958477
ridiculus mus	Vestibulum ac est lacinia nisi venenatis tristique...	-100.319738
congue	Aenean lectus. Pellentesque eget nunc. Donec quis ...	4.987737

Number of products 13-17(product_id) in stock in each warehouse

SELECT product_location.product_id, product_location.warehouse_id, SUM(product_location.quantity) AS total_quantity

FROM product_location

WHERE product_location.product_id BETWEEN 13 AND 17

GROUP BY product_location.warehouse_id, product_location.product_id

ORDER BY product_location.product_id

✔ Showing rows 0 - 12 (13 total, Query took 0.0071 seconds.) [product_id: 13... - 17...]

```
SELECT product_location.product_id, product_location.warehouse_id, SUM(product_location.quantity) AS total_quantity FROM product_location WHERE product_location.product_id BETWEEN 13 AND 17 GROUP BY product_location.warehouse_id, product_location.product_id ORDER BY product_location.product_id
```

Profiling [Edit inline] [Edit] [

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

+ Options

product_id	warehouse_id	total_quantity
13	1	6044
13	2	5242
13	3	7773
14	6	11327
14	3	14767
15	4	983
15	9	7266
16	2	22033
16	5	6039
16	10	2362
17	1	7360
17	7	6574
17	9	5499

Name of every supplier with that has made a shipment to warehouse 3 in 2018

SELECT supplier.supplier_name

FROM supplier

INNER JOIN supplier_shipment

ON supplier.supplier_id = supplier_shipment.supplier_id

INNER JOIN shipment_details

ON supplier_shipment.shipment_id = shipment_details.shipment_id

WHERE supplier_shipment.shipment_date BETWEEN '2018-01-01' AND '2018-12-31'

AND supplier_shipment.warehouse_id = 3

GROUP BY supplier.supplier_id

✔ Showing rows 0 - 16 (17 total, Query took 0.0161 seconds.)

```
SELECT supplier.supplier_name FROM supplier INNER JOIN supplier_shipment ON supplier.supplier_id = supplier_shipment.supplier_id INNER JOIN shipment_details ON supplier_shipment.shipment_id = shipment_details.shipment_id WHERE supplier_shipment.shipment_date BETWEEN '2018-01-01' AND '2018-12-31' AND supplier_shipment.warehouse_id = 3 GROUP BY supplier.supplier_id
```

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

supplier_name
Kulas, Nienow and Hegmann
King, Toy and Hickle
Schneider, Torp and D'Amore
Little-Auer
O'Connell Inc
Brakus-Kertzmann
Bode-Huels
Cruickshank-Blick
Brekke Group
O'Reilly-Goldner
Kiehn-Greenfelder
Bradtke-Barrows
Goldner Group
Larkin-Pfannerstill
Dooley-Kertzmann
Rutherford and Sons
Boyle, Legros and Roberts

Insert new customer and order of 55 units of product 25 at current price into the database

INSERT INTO customer(first_name, last_name, dob, email_address, phone_number) VALUES ('Toast', 'Cactus', '1975-11-20', 'toast@binkmail.com', 7057774242);

INSERT INTO customer_order(customer_id, order_date, shipping_address) VALUES (LAST_INSERT_ID(), '2018-12-04', '234 Cactus Ave.');

INSERT INTO order_details(order_id, product_id, quantity, sale_price)

SELECT order_id, product_id, 55, unit_price

FROM

(SELECT product_id, unit_price FROM product WHERE product_id = 25) AS product_id_price,

(SELECT order_id from customer_order WHERE order_id = LAST_INSERT_ID()) AS last_order_id;

1 row inserted.
Inserted row id: 102 (Query took 0.0079 seconds.)

```
INSERT INTO customer(first_name, last_name, dob, email_address, phone_number) VALUES ('Toast', 'Cactus', '1975-11-20', 'toast@binkmail.com', 7057774242)
```

[Edit inline] [Edit] [Create PHP code]

1 row inserted.
Inserted row id: 1002 (Query took 0.0043 seconds.)

```
INSERT INTO customer_order(customer_id, order_date, shipping_address) VALUES (LAST_INSERT_ID(), '2018-12-04', '234 Cactus Ave.')
```

[Edit inline] [Edit] [Create PHP code]

1 row inserted.
Inserted row id: 1002 (Query took 0.0059 seconds.)

```
INSERT INTO order_details(order_id, product_id, quantity, sale_price) SELECT order_id, product_id, 55, unit_price FROM (SELECT product_id, unit_price FROM product WHERE product_id = 25) AS product_id_price, (SELECT order_id from customer_order WHERE order_id = LAST_INSERT_ID()) AS last_order_id
```

[Edit inline] [Edit] [Create PHP code]

	customer_id	first_name	last_name	dob	email_address	phone_number
Edit Copy Delete	101	Toast	Cactus	1975-11-20	toast@binkmail.com	7057774242

	order_id	customer_id	order_date	shipping_address
Edit Copy Delete	1002	102	2018-12-04	234 Cactus Ave.

	order_id	product_id	quantity	sale_price
Edit Copy Delete	1002	25	55	67.36

Scaling Efficiency

Databases can be made more efficient for scale in a number of ways. One technique is denormalization, where a database is changed from the normal forms in some way to improve efficiency and reduce the need for joins and other costly operations[1] [2].

One form of denormalization is collapsing relationships by merging tables together. This eliminates the need to join tables when searching as well as reducing foreign keys which can significantly improve performance in larger databases but also adds redundancy as data will be duplicated if the relationship between the datas is not one-to-one. It is most effective in systems where data is not updated frequently as updates need to be done multiple times if there is redundant data.

Another type of denormalizations is partitioning tables. Tables are split horizontally or vertically to minimize the amount of data being searched. This is effective is certain portions of the data are more frequently used.

Adding redundant data is another way to improve performance in some situations. If an attribute of one table is frequently accessed by joining with another table it may be worthwhile adding a copy to the second table to eliminate the need to join, even if it results in duplicate data. Similarly, adding derived attributes as columns can reduce the need for calculations if the same derived attributes are being accessed repeatedly.

[1] https://www.sciencedirect.com/science/article/pii/S0167923604003021

[2] https://pdfs.semanticscholar.org/2c79/069c01ba8d598f32e61fe367ef6d261a0cb4.pdf

PHP Script to export to CSV

See export_to_csv.php

```php
<?php
// connect to database
$con=mysqli_connect("localhost","root","","inventory_management");
if (mysqli_connect_errno($con))
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
else
{
    echo "Success; you are connected"; ?> <br> <?php
}

// list of all table names
$tableNames = array("customer", "customer_order", "order_details", "product", "product_location", "warehouse", "supplier", "supplier_shipment", "shipment_details");
$numTables = count($tableNames);
// iterate through tables and export each table as a csv file
for ($i = 0; $i < $numTables; $i++)
{
    $sql = "SELECT * FROM {$tableNames[$i]}
    INTO OUTFILE '{$tableNames[$i]}.csv'
    FIELDS TERMINATED BY ','
    OPTIONALLY ENCLOSED BY '\"'
    LINES TERMINATED BY '\\n'";

    if (mysqli_query($con,$sql))
    {
        echo "Data submitted successfully";
    }
    else
    {
        echo "Error submitting the data into the backend " . mysqli_error($con);
    }
}
// close connection
mysqli_close($con);
?>
```