```
-- to drop those tables that have been created earlier
DROP TABLE Customers CASCADE;
DROP TABLE Orders;
DROP TABLE Items;
DROP TABLE Suppliers CASCADE;
DROP TABLE Places;
DROP TABLE Has;
DROP TABLE Supplied;
DROP TABLE Shipped;
```

I/ CREATE TABLE with constraints (Primary key, foreign key, attribute constraint, and tuple constraint)

```
CREATE TABLE Customers (
    id int NOT NULL,
    name varchar(20),
    phone varchar(16),
    email varchar(30),
    addr varchar(30),
    primary key (id)
  );

  CREATE TABLE Orders(
    Order_numb int NOT NULL,
    Order_date varchar, --YYYY-MM-DD
    Tracking varchar(20),
    amount DECIMAL(6,2) check (amount > 0.0 and amount < 3000),
    paymentType VARCHAR(30) check (paymentType = 'Credit Card' or paymentType = 'PayPal'),
    transaction VARCHAR(20),
    PRIMARY KEY (Order_numb)
  );

  CREATE TABLE Items(
    id int NOT NULL,
    Cost DECIMAL(6,2),
    Name VARCHAR(100),
    Discount Decimal(4,2),
    Description varchar(200),
    price DECIMAL(6,2),
    PRIMARY KEY(id),
    Check(price > cost + price * discount)
  );

  CREATE TABLE Suppliers(
    ID INT NOT NULL,
    Name VARCHAR(20),
    Addr VARCHAR(20),
    phone VARCHAR(20),
    PRIMARY KEY(ID)
  );

  CREATE TABLE Places(
    Cus_ID INT references Customers(id) on update cascade,
    Order_numb INT PRIMARY KEY
  );

  CREATE TABLE Has(
    Order_numb int,
    item_ID int,
    Quantity int check(quantity > 0),
    PRIMARY KEY (Order_numb, item_ID)
  );

  CREATE TABLE Supplied(
    item_ID int PRIMARY KEY,
    Supplier_ID int references Suppliers(ID) on delete set null
  );

  CREATE TABLE Shipped(
    Order_numb int PRIMARY KEY,
    Supplier_ID int,
    Foreign Key (supplier_ID) references Suppliers(ID) on delete set null
  );
```

II/ Insert value into tables

```
INSERT INTO Customers VALUES (1234, 'Sally', '870-891-9381', 'sally1@gmail.com',
'Jonesboro');
INSERT INTO Customers VALUES (2893, 'John', '609-287-1822', 'johnsonbaby@gmail.com',
'Memphis');
INSERT INTO Customers VALUES (8912, 'Linda', '831-849-2874', 'linda1932@gmail.com', 'Egg
Harbor Township');
INSERT INTO Customers VALUES (7719, 'Billy', '762-991-4211', 'billy@gmail.com',
'Philadelphia');
INSERT INTO Customers VALUES (3081, 'James', '212-772-2144', 'james12@gmail.com',
'Austin');

INSERT INTO Orders VALUES (123000, '2017-09-01', '1ZE42F480209223788', 30, 'Credit Card',
'5527351');
```

```
INSERT INTO Orders VALUES (489200, '2017-02-11', '1ZE43H382309223801', 40, 'Credit Card',
'6087312');
INSERT INTO Orders VALUES (431030, '2017-05-30', '1ZE42F481435497788', 25, 'PayPal',
'5126435');
INSERT INTO Orders VALUES (129473, '2018-12-12', '1ZE42F097712438726', 100, 'PayPal',
'2308549');

INSERT INTO Items VALUES (9888, 10, 'Big Girls Ribbed Sweater Dress', 0, 'cute and casual
stylish sweater dress', 25);
INSERT INTO Items VALUES (1992, 12, 'Crochet-Trim Bell-Sleeve Dress', 0.10, 'Pretty crochet
lace trim', 30);
INSERT INTO Items VALUES (4801, 40, 'Women''s Saltwater Duck Booties', 0, 'Let this
waterproof style take you where others
 can''t go', 100);
INSERT INTO Items VALUES (2848, 10, 'Around-Town Flip-Top Mittens', 0, 'A buttoned flip-top
lends around-town versatility', 40);
INSERT INTO Items VALUES (3892, 100, 'BL770 Blender & Food Processor', 0.15, 'From mixing
dough to making single-serve smoothies', 240);
INSERT INTO Items VALUES (1277, 20, '3-Qt. Soup Pot with Lid', 0, 'A classic look with
contemporary performance', 40);
INSERT INTO Items VALUES (8921, 20, 'Tanjun Casual Sneakers from Finish Line', 0.10, 'Modern
and comfortable', 45);
INSERT INTO Items VALUES (8943, 35, 'Free Run 2018 Running Sneakers', 0.10, 'Features an
upgraded sole design for a natural feel', 70);

INSERT INTO Suppliers VALUES (890, 'Ninja', 'Los Angeles', '981-378-2861');
INSERT INTO Suppliers VALUES (134, 'Under Armour', 'Pittsburgh', '217-972-9910');
INSERT INTO Suppliers VALUES (367, 'Belgique', 'Bishop', '743-219-8475');
INSERT INTO Suppliers VALUES (772, 'Nike', 'Campbell', '972-843-2854');
INSERT INTO Suppliers VALUES (471, 'HM', 'Brea', '874-987-8124');
INSERT INTO Suppliers VALUES (032, 'BestBuy', 'Little Rock', '609-432-4371');

INSERT INTO Places VALUES (2893, 489200);
INSERT INTO Places VALUES (3081, 129473);
INSERT INTO Places VALUES (8912, 431030);
INSERT INTO Places VALUES (2893, 123000);

INSERT INTO Has VALUES (489200, 2848, 1);
INSERT INTO Has VALUES (129473, 4801, 1);
INSERT INTO Has VALUES (431030, 9888, 1);
INSERT INTO Has VALUES (123000, 1992, 1);

INSERT INTO Supplied VALUES (8921, 772);
INSERT INTO Supplied VALUES (8943, 772);
```

```sql
INSERT INTO Supplied VALUES (9888, 890);
INSERT INTO Supplied VALUES (1992, 134);
INSERT INTO Supplied VALUES (4801, 367);
INSERT INTO Supplied VALUES (2848, 134);
INSERT INTO Supplied VALUES (3892, 471);
INSERT INTO Supplied VALUES (1277, 032);

INSERT INTO Shipped VALUES (123000, 134);
INSERT INTO Shipped VALUES (489200, 772);
INSERT INTO Shipped VALUES (431030, 890);
INSERT INTO Shipped VALUES (129473, 367);
```

III/ Queries

```sql
-- ============ 8 simple queries (similar to the examples below)
--          operators includes (and,or,not)
--          patterns

 -- SELECT ... FROM ... WHERE

 -- 1. Find all the customers' names
SELECT name FROM Customers;

-- 2. Find customer who has order number is 123000
SELECT Cus_ID FROM Places WHERE Order_numb = 123000;

-- 3. Find all the products ID that supplied by supplier_ID 772
SELECT item_ID FROM Supplied WHERE Supplier_ID = 772;

-- 4. Find the price of 3-Qt. Soup Pot with Lid
SELECT price FROM items WHERE name = '3-Qt. Soup Pot with Lid';

-- 5. Find tracking number for order 129473
SELECT tracking FROM Orders where order_numb = 129473;

-- 6. Find the Orders that use PayPal as its payment method and the amount is $25
SELECT Order_numb FROM Orders WHERE paymentType = 'PayPal' AND amount = 25;

-- 7. Find the customers that their phone number's area code is 870;
SELECT ID FROM Customers WHERE phone LIKE '870%';

-- 8. Find the items that offer no discount or price less than 30
SELECT name FROM iTEMS where discount = 0 OR price < 30;

-- ============= 6 Multirelation queries (two or more relations
--                             in the FROM-clause)
-- (similar to the examples below)

-- 9. Find the addresses, and names for those who ordered 'Around-Town Flip-Top Mittens'
SELECT  addr, name FROM Customers, Places where id = Cus_ID AND Order_numb = (Select
order_numb FROM has
WHERE item_ID = (SELECT ID FROM Items WHERE name = 'Around-Town Flip-Top Mittens'));

-- 10. Find the orders that shipped by Nike
SELECT order_numb FROM Shipped, Supplier WHERE Supplier_ID = ID and name = 'Nike';

-- using operators and or not

-- 11. Find the Order that shipped by Ninja and paid by credit Card
SELECT Orders.Order_numb FROM Orders, Supplier, shipped WHERE orders.paymentType =
'Credit Card' AND Supplier.Name = 'Ninja'
AND Orders.Order_numb = shipped.Order_numb AND Supplier.ID = Shipped.Supplier_ID;

 -- 12.Find the phone number of customer who placed order 431030
 SELECT addr from Customers, Places where id = cus_ID AND Order_numb = 431030;

 -- 13. Find the items that its supplier is 890 or located at Los Angeles
 SELECT items.Name FROM Items, Supplied, Suppliers where Items.ID = Supplied.Item_ID AND
Supplied.Supplier_ID = Suppliers.ID
 AND (suppliers.ID = 890 OR suppliers.Addr = 'Los Angeles');

 -- 14. Find the order that placed by Linda
 SELECT Order_numb FROM Places, Customers WHERE Customers.ID = Places.Cus_ID AND
Customers.name = 'Linda';

-- ============= 6 Subqueries like below

 -- FROM (subquery)

 --15. have subquery in FROM clause
 -- WHERE in (subquery)
 --16. have subquery with keyword "IN"
 --Find the order that not using PayPal as its payment method
 SELECT order_numb FROM Orders WHERE Order_numn NOT IN (SELECT Order_numb FROM
Orders WHERE paymentType = Paypal);

 --17. EXISTS (e.g. unique, all)
 -- Find the most cheapest items sold in store
 SELECT name FROM iTEMS i1 WHERE not EXISTS (select * from items where i1.price > price);
 --18. ANY
 -- Find the items that return profit of $30 or more.
 SELECT name, cost, price from Items where id = any (select id from items where (price * (1 -
discount) - cost) >= 30);
 --19. ALL
 -- Find the most expensive items sold in store
 SELECT name FROM items WHERE price >= ALL (SELECT price FROM items);

 --20. Find the supplier who shipped order 129473
 SELECT * from suppliers where id = (select supplier_ID from shipped where order_numb =
129473);

-- ===============  5 SQL-statements using union, intersect, difference (except)

 -- 21. Find the items are not sold ( no one ordered them)
 (SELECT id, name from items) except (SELECT id, name from items where ID in (select item_ID
from has where quantity >= 1));

 -- 22. Find the items that costs $10 or $20
 (SELECT name, cost from items where cost <= 10) union (select name, cost from items where
cost >= 100);

 -- 23. Find the orders placed during 2017 and paid by Credit Card
 (SELECT Order_numb from Orders where Order_date like '2017%') intersect (SELECT
Order_numb from Orders
    where paymentType = 'Credit Card');

 -- 24. Find the customers that doesn't place any orders
 (SELECT name from Customers) except (SELECT name from customers where id in (select
cus_ID from places));

 -- 25. Find the items that cost $20 or under and offer discount
 (SELECT id, name from items where cost < 20) intersect (select id, name from items where
discount > 0);

-- ==============5 SQL-statements using Join ==================================
 -- using CROSS JOIN, NATURAL JOIN, THETA JOIN (INNER JOIN)
 -- 26. List the details of all orders includes their item_ID and Quantity
 SELECT * from orders natural join has;

 -- 27. List all the customers with their order number.
 Select * from Customers JOIN places on id = Cus_ID;
```

-- 28. Find the customers whose their items are shipped by Nike 772
select name from customers join places on id = cus_ID where order_numb = (select order_numb from shipped where supplier_ID = 772);

-- 29. List all the suppliers along with its products
SELECT * FROM suppliers join supplied on id = supplier_ID;
-- 30. List all the orders that shipped by their suppliers;
SELECT * FROM HAS Natural JOIN shipped;

-- ============================== Aggregate Functions
-- MAX, MIN, SUM, AVG, COUNT

-- using GROUP BY

-- using HAVING

-- 31. Find the most expensive items that store offers
SELECT name, price from iTEMS where price = (SELECT MAX(price) FROM Items);
-- 32. Find the average of all transactions processed through PayPal
SELECT AVG(amount) FROM Orders GROUP BY paymentType HAVING paymentType = 'PayPal';
-- 33. Find the sum of all orders placed in 2017
SELECT SUM(amount) From orders where Order_date like '2017%';
-- 34.Count total orders were placed during 2017
SELECT COUNT * from Orders where order_date like '2017%';
-- 35. Find the minimum order
SELECT order_numb, amount from Orders where amount = (SELECT Min(amount) from Orders);

IV/ Database Modification

-- 36. Insert a new order
INSERT INTO Orders VALUES (289147, '2018-07-01', '1ZE42F086192935636', 100, 'PayPal', '2881740');

-- 37. Insert an item into an order
INSERT INTO has (select order_numb, id, 3 from Orders, Items where order_numb = 123000 and id = 2848);

-- 38. Delete an order
DELETE FROM Orders WHERE Order_numb = 289147;
-- 39. Delete an item from an order
DELETE FROM has where Order_numb = 123000 AND item_ID = 2848;
-- 40. Update price of an item
UPDATE Items set price = 50 where id = 1277;

```
  -- 41. Update BestBuy's contact number
  UPDATE supplier set phone = '609-432-4982' where name = 'BestBuy';
```

V/ Create View

```
-- 42. create view of sale during 2017
  CREATE View 2017Sale AS (select * from Orders where Order_date Like '2017%');
```

VI/PSM

```
-- 43. trigger
  CREATE trigger CustomersTrig
  After INSERT ON place
  referencing
  new row as nnn
  for each row
  when Cus_ID NOT IN (select id from Customers)
  Insert into Customers(id) VALUES (nnn.cus_iD);

-- 44. PSM : evaluate the profitability of items
  CREATE Function HighProfit (in a integer)
  return varchar(15)
  declare profit DECIMAL(6,2);
  Begin
  set profit = (select (price * (1 - discount) - cost) / cost from items where id = a);
  if profit >= 1.50 Then return 'extremely high profit';
  elseif profit >= 80 then return 'high profit';
  elseif profit >= 40 then return 'profitable';
  else return 'Should consider to stop selling this product';
  end if;
  end;
```

VII/ Relational Algebra – Functional Dependencies

--46. one relational algebra

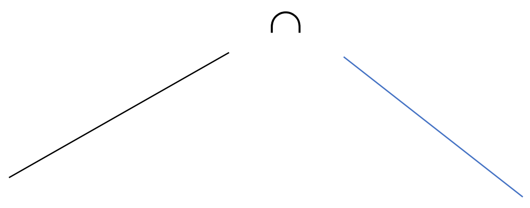
Find the Orders that use PayPal as its payment method and the amount is $25

SELECT Order_numb FROM Orders WHERE paymentType = 'PayPal' AND amount = 25;

Trans25PayPal := $\pi_{\text{Order\_numb}}(\sigma_{\text{paymentType = “PayPal” and amount = 25}}(\text{Orders}))$

--47. one relational algebra tree

Trans25PayPal := $\sigma_{\text{paymentType = “PayPal” and amount = 25}}(\text{Orders})$



$$\begin{tikzcd} \pi_{\text{Order\_numb}} \arrow[d, dash] & \pi_{\text{Order\_numb}} \arrow[d, dash] \\ \sigma_{\text{paymentType = “PayPal”}} \arrow[d, dash] & \sigma_{\text{amount =}}\ 25 \arrow[d, dash] \\ \text{Order} & \text{Order} \end{tikzcd}$$

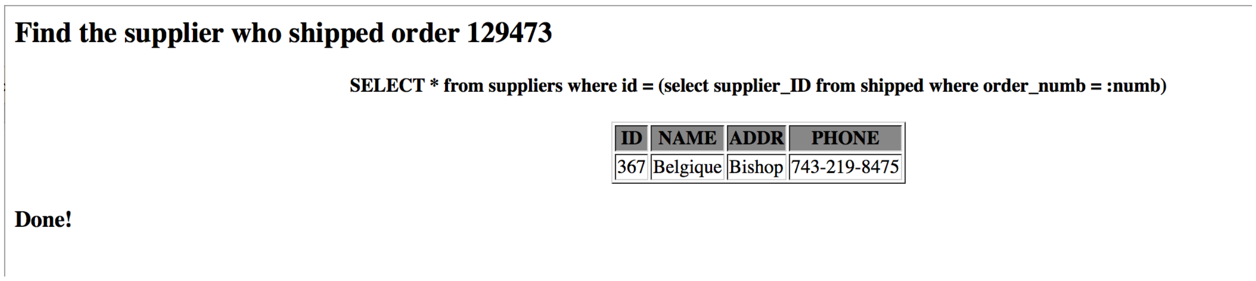
--48. functional dependencies for each table
Customers: id -> name, phone, email, addr
Orders: Order_numb -> Order_date, Tracking, Amount, PaymentType, Transaction
Items: id -> Cost, Name, Discount, Description, Price
Suppliers: ID -> name, addr, phone
Places: Order_numb -> Cus_ID
Has: Order_numb, item_ID -> Quantity
Supplied: item_ID -> Supplier_ID
Shipped: Order_numb -> Supplier_ID

All tables follow BCNF.

<u>Interface1:</u>

```
<HTML>
<head>
<title> Which supplier? </title>
</head>
<body>
<?PHP
    $dsn="pgsql:host=localhost;dbname=minhtran.tran";   // data source name
    $dbuser='minhtran.tran';
    $password = 'ahihi123';

    $conn = new PDO($dsn, $dbuser, $password);

    if (!$conn)
    {
        echo "Could not connect!!!!\n";
        exit;
    }
    echo "<h2> Find the supplier who shipped order 129473 </h2> \n";
```

```
	$query = "SELECT * from suppliers where id = (select supplier_ID from shipped where
order_numb = :numb)";
	echo "<h4 align=\"center\"> $query </h4> \n";

	//prepare the SQL statement
	$sqlquery=$conn->prepare($query, array(PDO::ATTR_CURSOR =>
PDO::CURSOR_FWDONLY));
	// execute the SQL statement
	$sqlquery->execute(array(':numb' => 129473));

	// get the results of the sql statement
	if ($row = $sqlquery->fetch(PDO::FETCH_ASSOC))
	{
		echo "<table border=\"1\" align=\"center\">\n";  //table
		echo "<tr>";
		foreach ($row as $key=>$value)
			echo "<th bgcolor=\"#8B8A8A\">".strtoupper($key)."</th> ";
		echo "</tr>\n";
		do {
			echo "<tr>";
			foreach ($row as $key => $value)
			{
				//echo "$key: $value ";
				echo "<td>" . $row["$key"] . "</td> ";
			}
			echo "</tr>\n";
		} while($row = $sqlquery->fetch(PDO::FETCH_ASSOC));
		echo "</table>";
	}

?>

<h3>Done!</h3>

</body>
</html>
```



Find the supplier who shipped order 129473

SELECT * from suppliers where id = (select supplier_ID from shipped where order_numb = :numb)

ID	NAME	ADDR	PHONE
367	Belgique	Bishop	743-219-8475

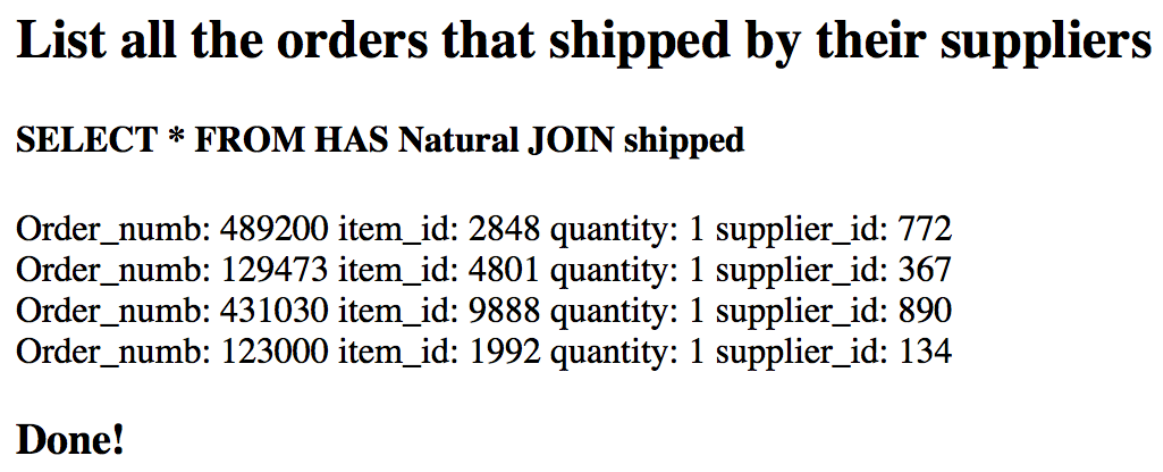
Done!

http://147.97.156.233/~minhtran.tran/OnlineStore2.php

Interface2:

```
<HTML>
<head>
<title> Details of Suppliers </title>
</head>
<BODY>
<?PHP
$dsn="pgsql:host=localhost;dbname=minhtran.tran";   // data source name
$dbuser='minhtran.tran';
$password = 'ahihi123';

$conn = new PDO($dsn, $dbuser, $password);

        if (!$conn)
        {
                echo "Could not connect!!!!\n";
                exit;
        }
        echo "<h2> List all the orders that shipped by their suppliers </h2> \n";
        $query = "SELECT * FROM HAS Natural JOIN shipped";
        echo "<h4> $query </h4> \n";

        $sqlquery=$conn->prepare($query);
        $sqlquery->execute();

while($row = $sqlquery->fetch())
{
        {
                echo "Order_numb: $row[0]  item_id: $row[1]  quantity: $row[2]  supplier_id:
$row[3]" ;
                echo "<br />\n";
        }
}

$query=null;

?>
```

```
<H3>Done!</H3>

</BODY>
</HTML>
```



List all the orders that shipped by their suppliers

SELECT * FROM HAS Natural JOIN shipped

```
Order_numb: 489200 item_id: 2848 quantity: 1 supplier_id: 772
Order_numb: 129473 item_id: 4801 quantity: 1 supplier_id: 367
Order_numb: 431030 item_id: 9888 quantity: 1 supplier_id: 890
Order_numb: 123000 item_id: 1992 quantity: 1 supplier_id: 134
```

Done!

http://147.97.156.233/~minhtran.tran/OnlineStore.php