DATABASE PROJECT

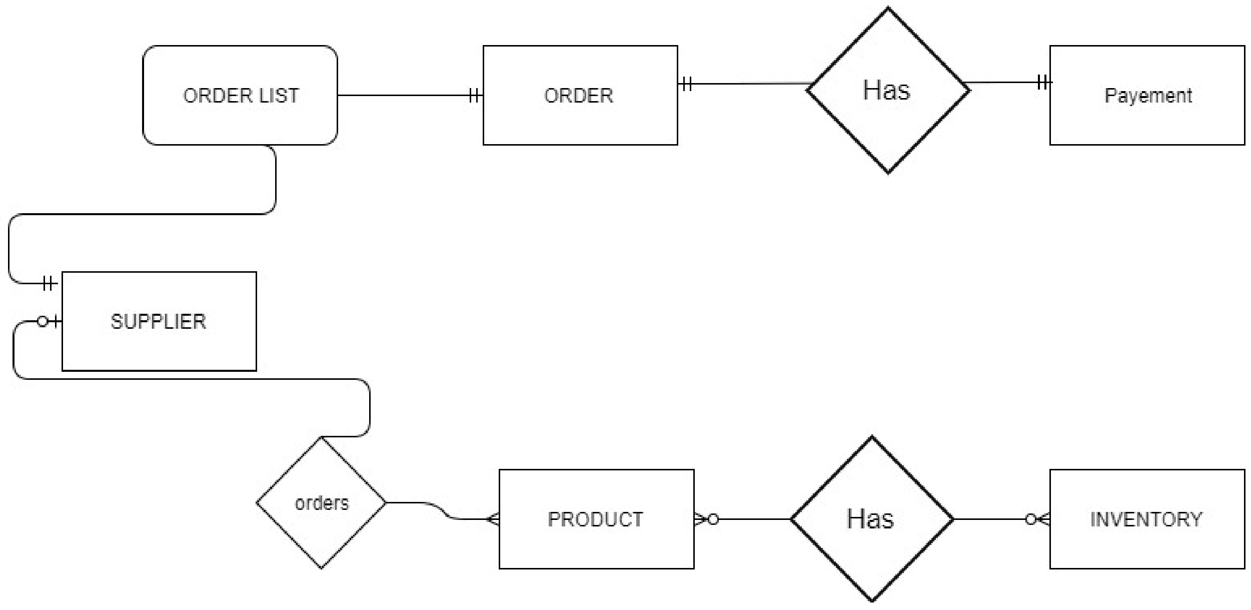
TITLE: - INVENTORY MANAGEMENT SYSTEM

Problem Statement:

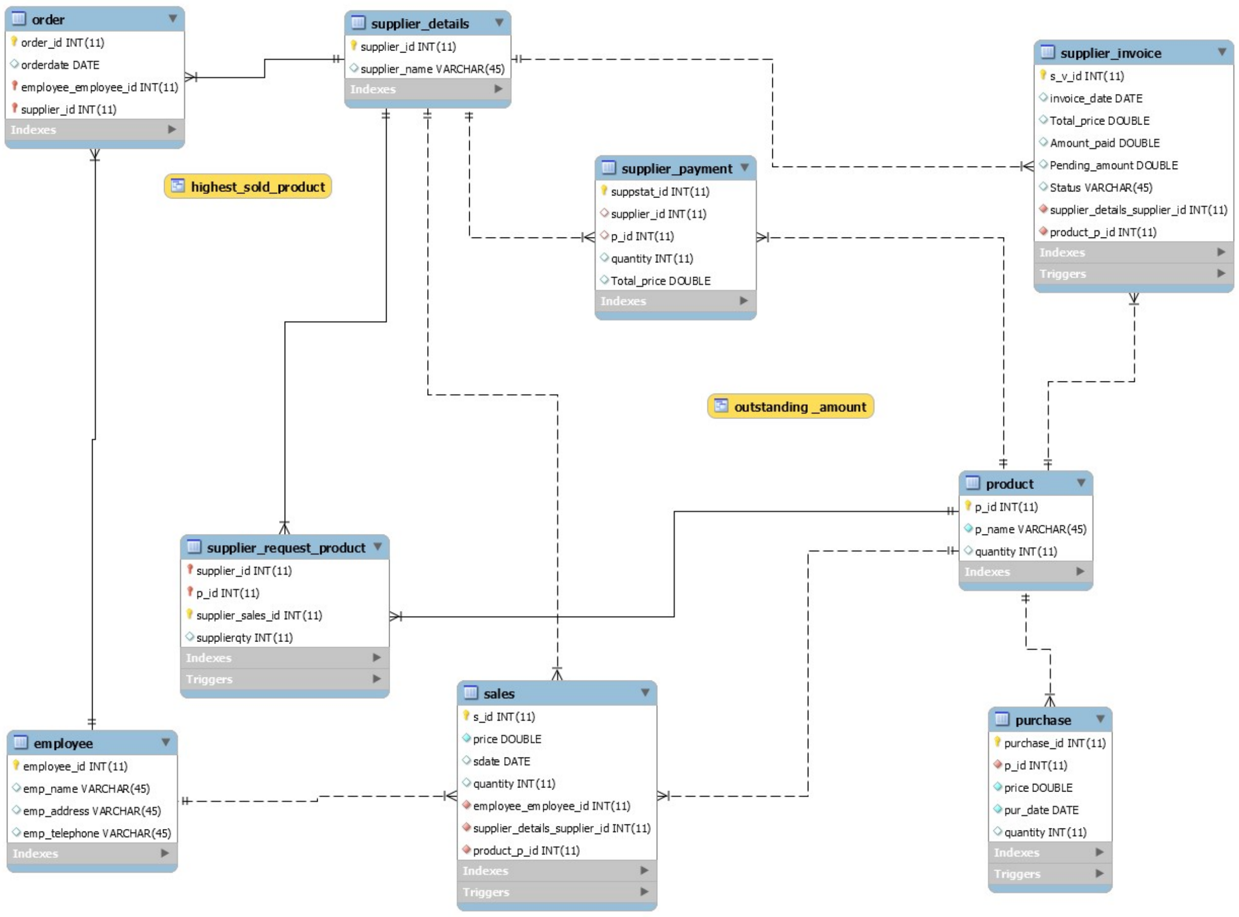
The inventory system is needed in order to maintain the record of the stock that is available. There should be a system which maintains database consisting of number of products ordered by the supplier, the price of that product, the availability of that product as and when the supplier demands the product from the manufacturer and to keep the report of no. of the product that supplier order as well as the total expense (profit) that has been done by the manufacturer in particular month or in a year.

This database will use trigger as well procedure. For example, whenever supplier orders from manufacturer, there should be procedure which decrements that particular product quantity from total quantity and trigger must be called whenever that particular order is not available in manufacturer inventory.

ER Diagram (Rough):

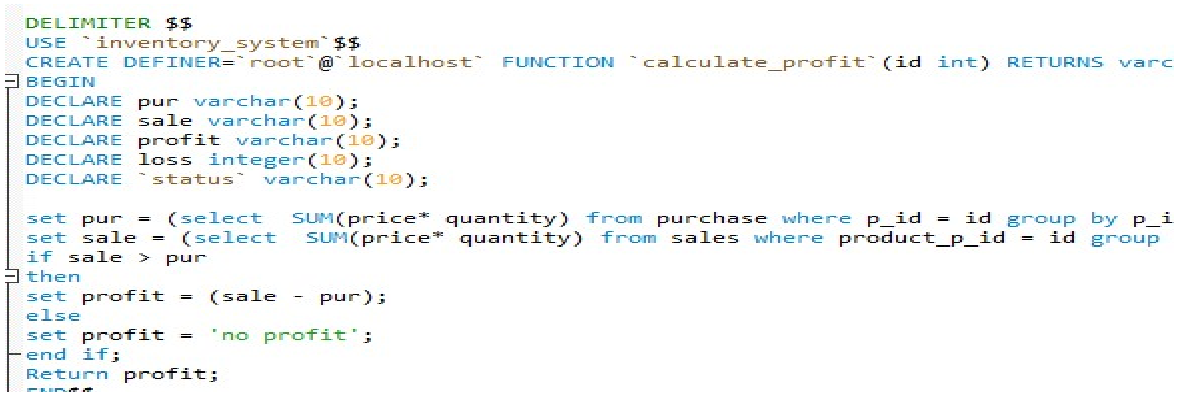


ER Diagram :



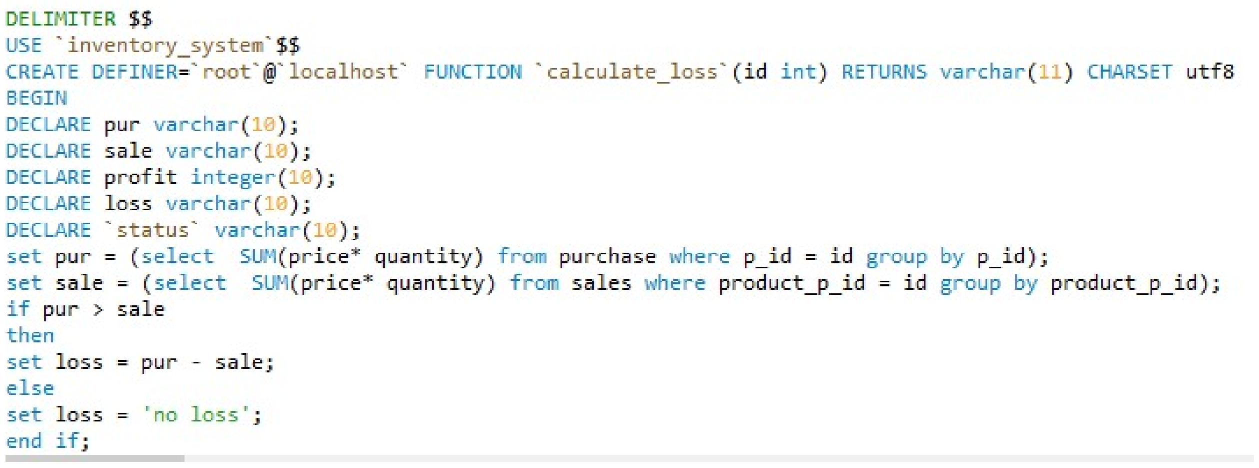
Functions:

1. To calculate the profit



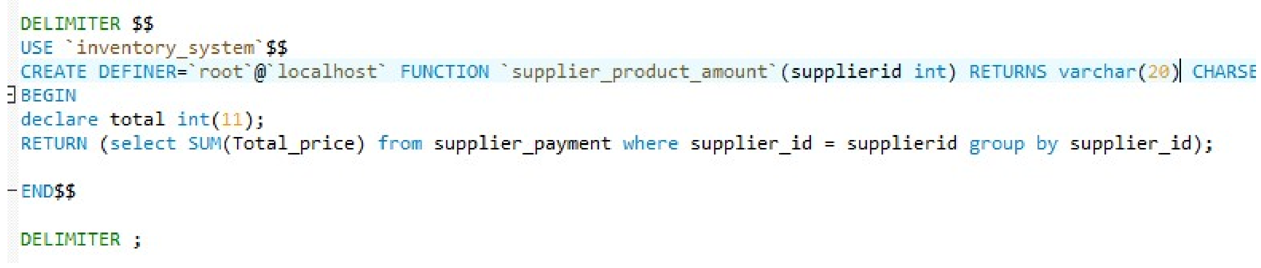
```
DELIMITER $$
USE `inventory_system`$$
CREATE DEFINER=`root`@`localhost` FUNCTION `calculate_profit`(id int) RETURNS varc
BEGIN
DECLARE pur varchar(10);
DECLARE sale varchar(10);
DECLARE profit varchar(10);
DECLARE loss integer(10);
DECLARE `status` varchar(10);

set pur = (select  SUM(price* quantity) from purchase where p_id = id group by p_i
set sale = (select  SUM(price* quantity) from sales where product_p_id = id group
if sale > pur
then
set profit = (sale - pur);
else
set profit = 'no profit';
end if;
Return profit;
```

2. To calculate the loss



```
DELIMITER $$
USE `inventory_system`$$
CREATE DEFINER=`root`@`localhost` FUNCTION `calculate_loss`(id int) RETURNS varchar(11) CHARSET utf8
BEGIN
DECLARE pur varchar(10);
DECLARE sale varchar(10);
DECLARE profit integer(10);
DECLARE loss varchar(10);
DECLARE `status` varchar(10);
set pur = (select  SUM(price* quantity) from purchase where p_id = id group by p_id);
set sale = (select  SUM(price* quantity) from sales where product_p_id = id group by product_p_id);
if pur > sale
then
set loss = pur - sale;
else
set loss = 'no loss';
end if;
```

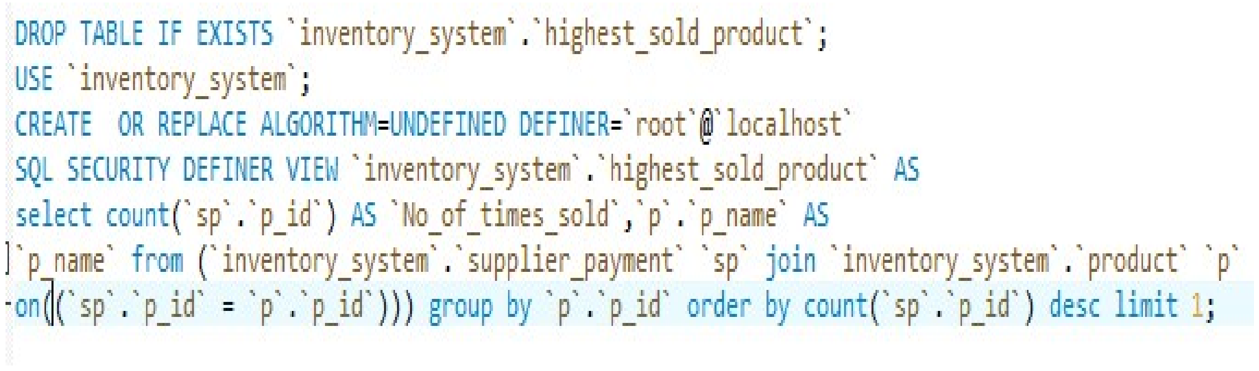
3. Total amount of the products purchased by the supplier



```
DELIMITER $$
USE `inventory_system`$$
CREATE DEFINER=`root`@`localhost` FUNCTION `supplier_product_amount`(supplierid int) RETURNS varchar(20) CHARSE
BEGIN
declare total int(11);
RETURN (select SUM(Total_price) from supplier_payment where supplier_id = supplierid group by supplier_id);

END$$

DELIMITER ;
```

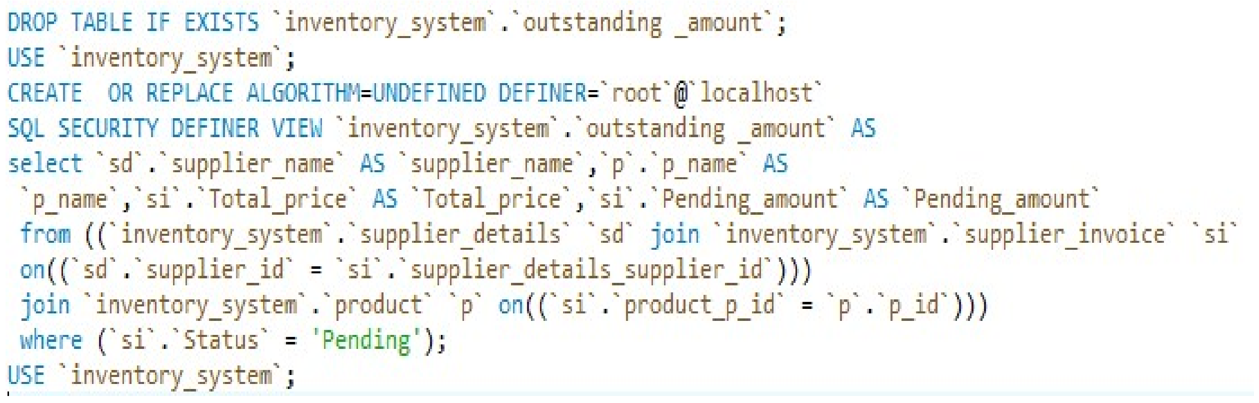
Views:

1. To view product sold the most



```
DROP TABLE IF EXISTS `inventory_system`.`highest_sold_product`;
USE `inventory_system`;
CREATE  OR REPLACE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost`
SQL SECURITY DEFINER VIEW `inventory_system`.`highest_sold_product` AS
select count(`sp`.`p_id`) AS `No_of_times_sold`,`p`.`p_name` AS
`p_name` from (`inventory_system`.`supplier_payment` `sp` join `inventory_system`.`product` `p`
on((`sp`.`p_id` = `p`.`p_id`))) group by `p`.`p_id` order by count(`sp`.`p_id`) desc limit 1;
```

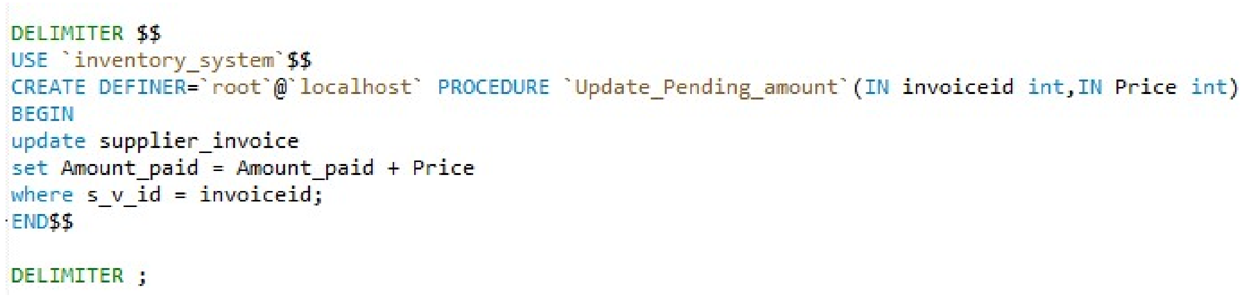
2. To view the outstanding amount of the suppliers



```
DROP TABLE IF EXISTS `inventory_system`.`outstanding _amount`;
USE `inventory_system`;
CREATE  OR REPLACE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost`
SQL SECURITY DEFINER VIEW `inventory_system`.`outstanding _amount` AS
select `sd`.`supplier_name` AS `supplier_name`,`p`.`p_name` AS
 `p_name`,`si`.`Total_price` AS `Total_price`,`si`.`Pending_amount` AS `Pending_amount`
 from ((`inventory_system`.`supplier_details` `sd` join `inventory_system`.`supplier_invoice` `si`
 on((`sd`.`supplier_id` = `si`.`supplier_details_supplier_id`)))
 join `inventory_system`.`product` `p` on((`si`.`product_p_id` = `p`.`p_id`)))
 where (`si`.`Status` = 'Pending');
USE `inventory_system`;
```

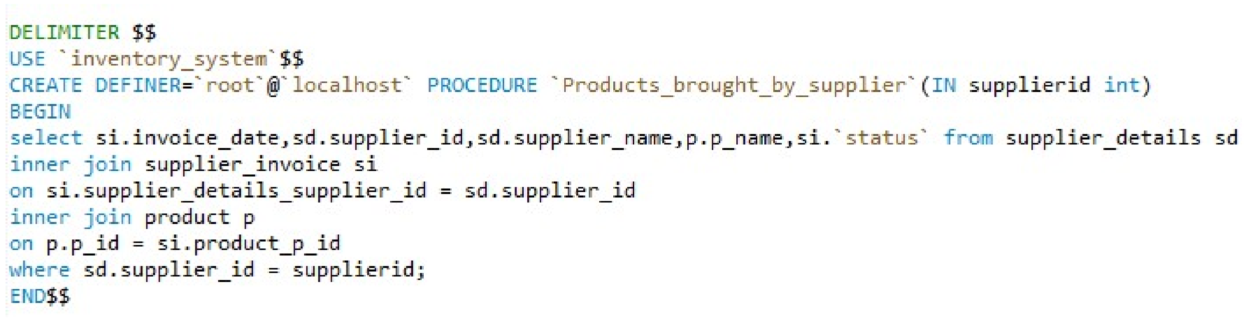
Stored Procedures:

1. To update the pending amount by mentioning the id and price



```
DELIMITER $$
USE `inventory_system`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `Update_Pending_amount`(IN invoiceid int,IN Price int)
BEGIN
update supplier_invoice
set Amount_paid = Amount_paid + Price
where s_v_id = invoiceid;
END$$

DELIMITER ;
```

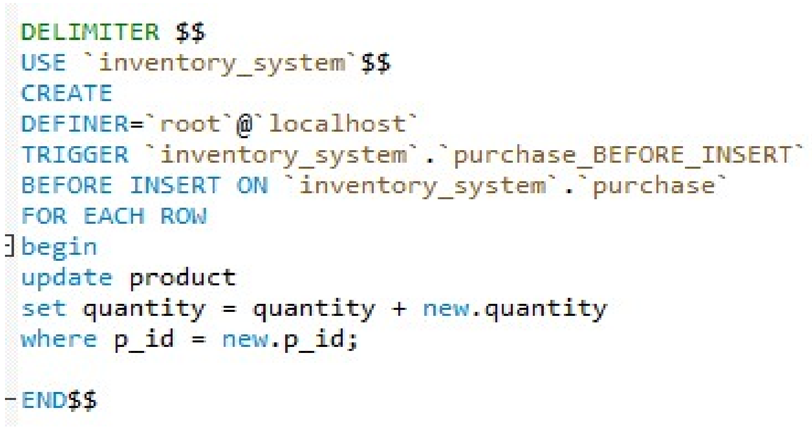
2. To view the products by the supplier



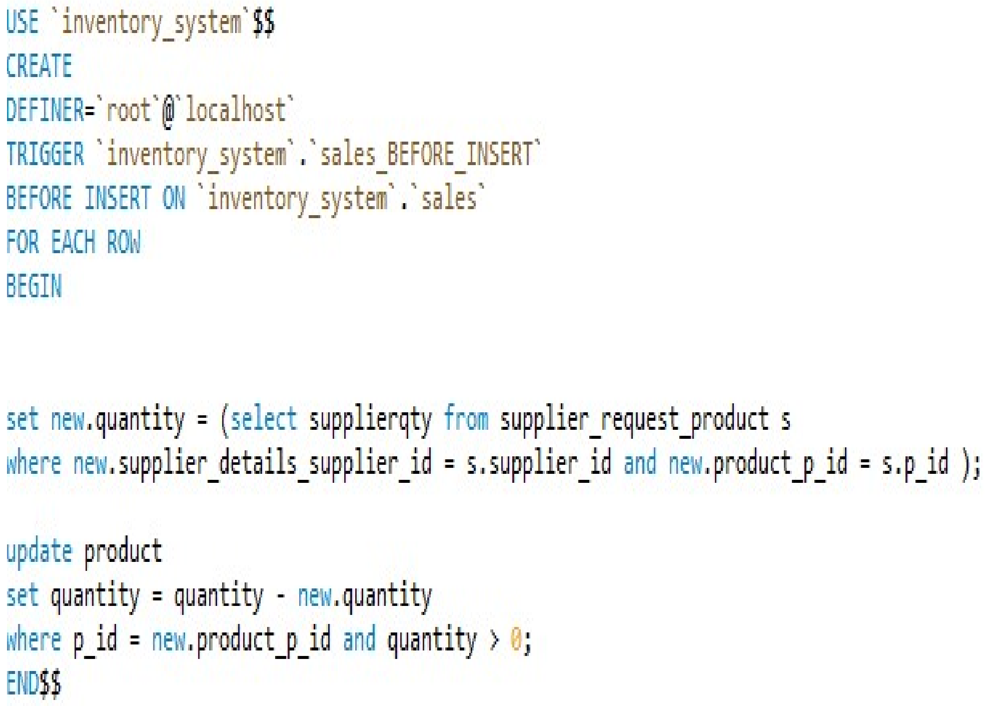
```
DELIMITER $$
USE `inventory_system`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `Products_brought_by_supplier`(IN supplierid int)
BEGIN
select si.invoice_date,sd.supplier_id,sd.supplier_name,p.p_name,si.`status` from supplier_details sd
inner join supplier_invoice si
on si.supplier_details_supplier_id = sd.supplier_id
inner join product p
on p.p_id = si.product_p_id
where sd.supplier_id = supplierid;
END$$
```

Triggers:

1. To update the quantity after purchase and sales of product

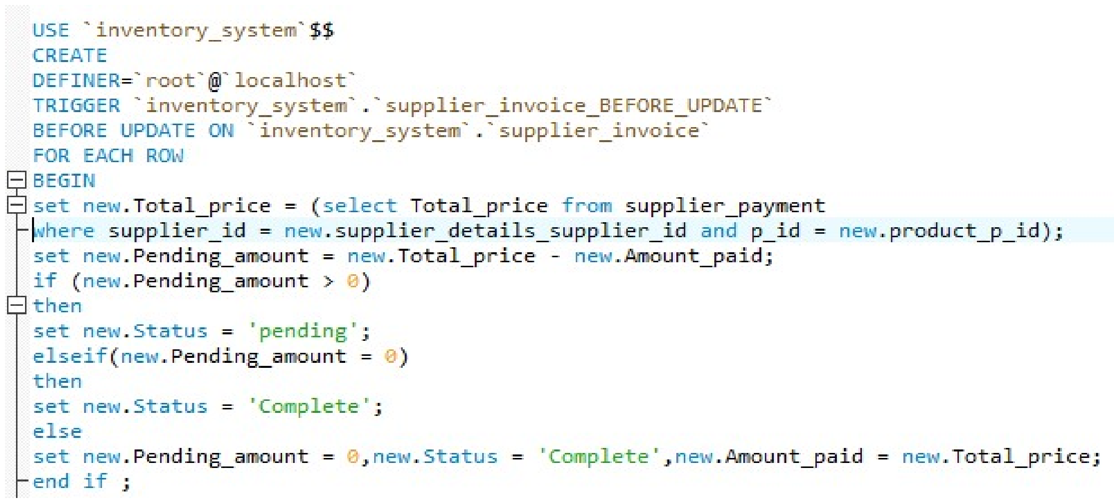


```
DELIMITER $$
USE `inventory_system`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `inventory_system`.`purchase_BEFORE_INSERT`
BEFORE INSERT ON `inventory_system`.`purchase`
FOR EACH ROW
begin
update product
set quantity = quantity + new.quantity
where p_id = new.p_id;

END$$
```



```
USE `inventory_system`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `inventory_system`.`sales_BEFORE_INSERT`
BEFORE INSERT ON `inventory_system`.`sales`
FOR EACH ROW
BEGIN


set new.quantity = (select supplierqty from supplier_request_product s
where new.supplier_details_supplier_id = s.supplier_id and new.product_p_id = s.p_id );

update product
set quantity = quantity - new.quantity
where p_id = new.product_p_id and quantity > 0;
END$$
```

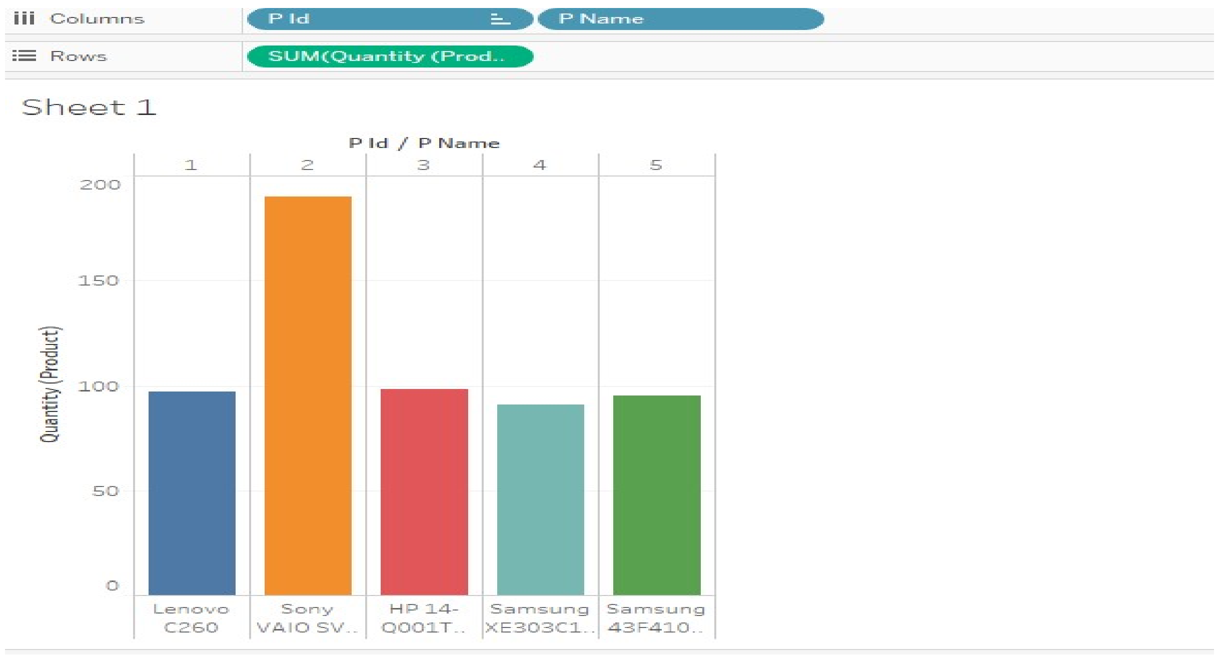
2. To update the supplier_invoice according to amount paid and whenever stored procedure (update_pending_amount) is executed



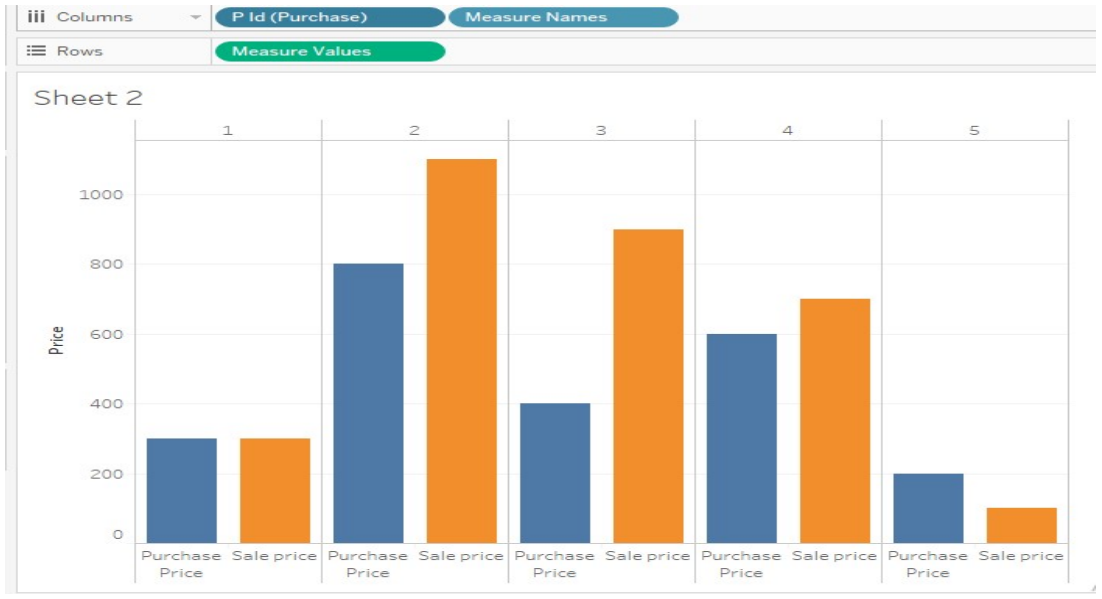
```
USE `inventory_system`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `inventory_system`.`supplier_invoice_BEFORE_UPDATE`
BEFORE UPDATE ON `inventory_system`.`supplier_invoice`
FOR EACH ROW
BEGIN
set new.Total_price = (select Total_price from supplier_payment
where supplier_id = new.supplier_details_supplier_id and p_id = new.product_p_id);
set new.Pending_amount = new.Total_price - new.Amount_paid;
if (new.Pending_amount > 0)
then
set new.Status = 'pending';
elseif(new.Pending_amount = 0)
then
set new.Status = 'Complete';
else
set new.Pending_amount = 0,new.Status = 'Complete',new.Amount_paid = new.Total_price;
end if ;
```

Graphs:

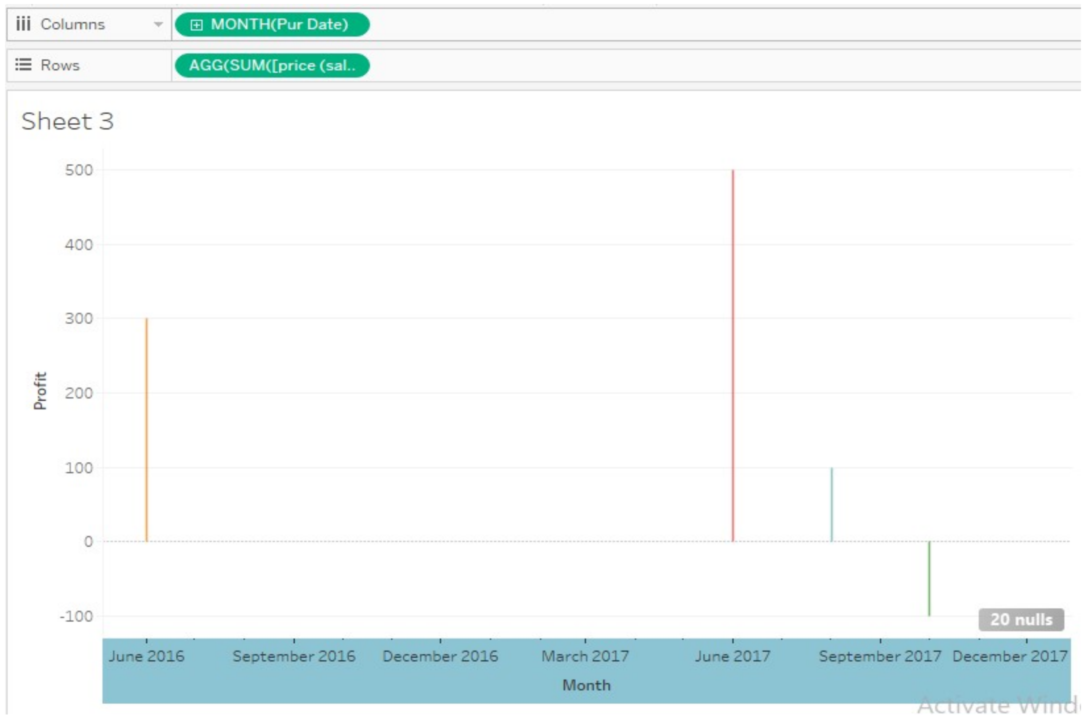
1. Total No. of Products available



2. Purchase price and sales price of each product

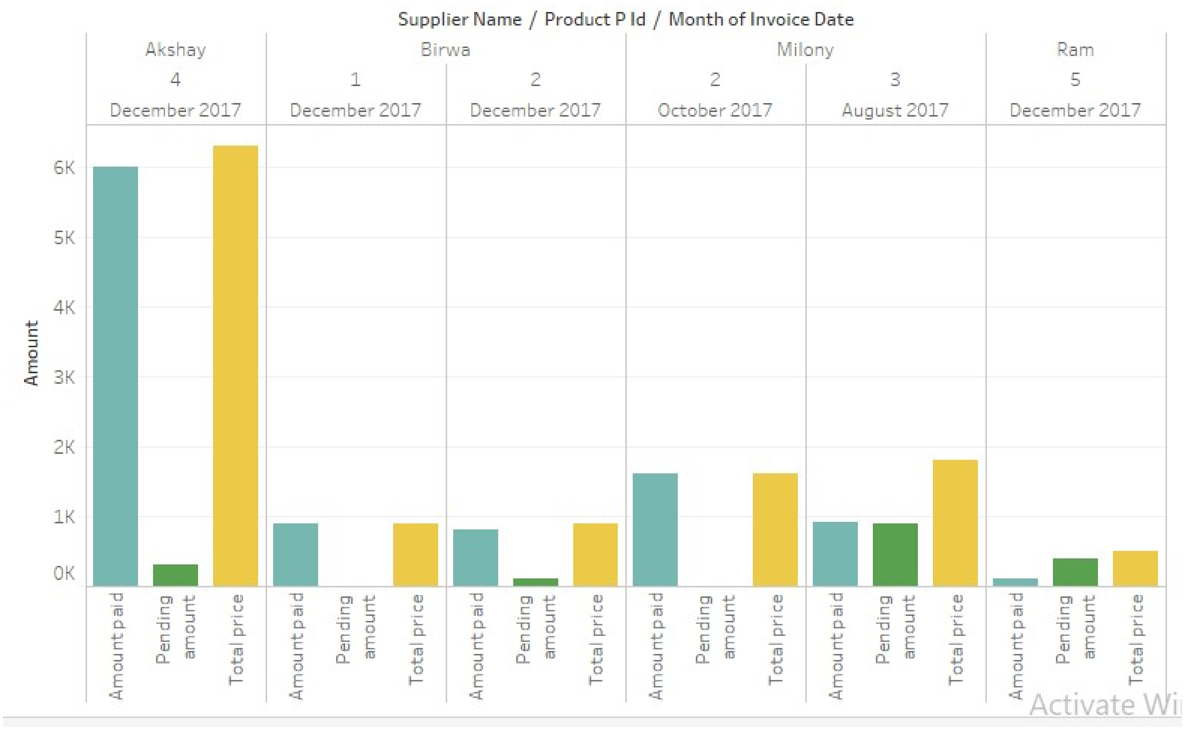


3. Calculating profit and loss of the product according to purchase price and sales price

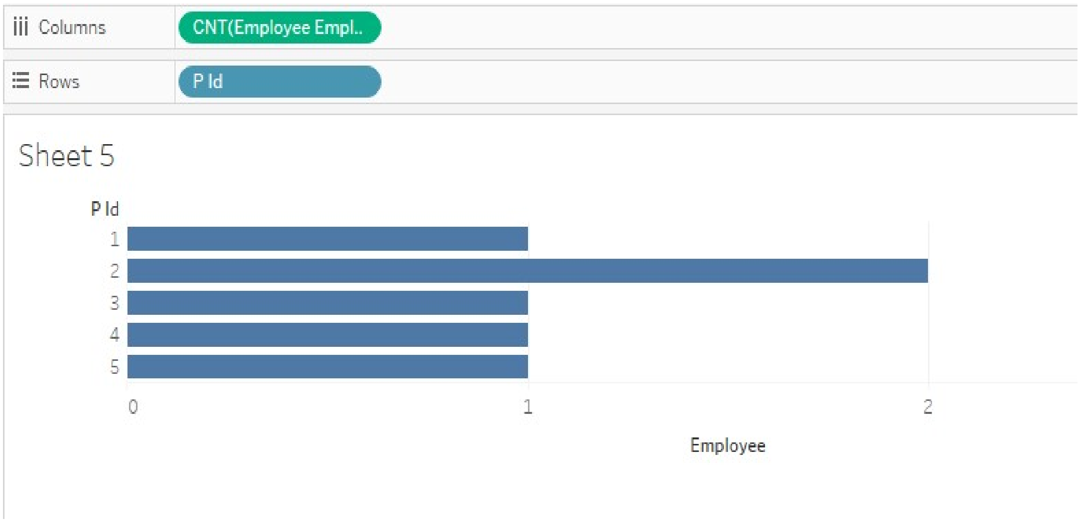


4. Supplier Billing Status

Sheet 4



5. The employee having maximum products


Columns
CNT(Employee Empl..
Rows
P Id
Sheet 5
P Id
1
2
3
4
5
0
1
2
Employee