CREATE TABLE CUSTOMERS  
(  
 CustNo numeric(5) PRIMARY KEY,  
 CustName varchar(20) NOT NULL,  
 Address1 varchar(50),  
 Address2 varchar(50),  
 City varchar(30),  
 State varchar(30),  
 PinCode varchar(10),  
 Phone varchar(30)  
);  
  
CREATE TABLE ORDERS  
(  
 OrdNo numeric(5) PRIMARY KEY,  
 OrdDate date,  
 ShipDate date,  
 CustNo numeric(5),  
 Address1 varchar(50),  
 Address2 varchar(50),  
 City varchar(30),  
 State varchar(30),  
 PinCode varchar(10),  
 Phone varchar(30),  
 FOREIGN KEY (CustNo) REFERENCES CUSTOMERS (CustNo),  
 CHECK (ShipDate >= OrdDate)  
);  
  
CREATE TABLE LINEITEMS  
(  
 OrdNo numeric(5) REFERENCES ORDERS (OrdNo),  
 ItemNo numeric(5) REFERENCES ITEMS (ItemNo),  
 Qty numeric(3),  
 Price numeric(8, 2),  
 DisRate numeric(4, 2) DEFAULT 0,  
 PRIMARY KEY (OrdNo, ItemNo),  
 CHECK (qty >= 1 AND DisRate >= 0)  
);  
  
CREATE TABLE ITEMS  
(  
 ItemNo numeric(5) PRIMARY KEY,  
 ItemName varchar(20),  
 Rate Numeric(8, 2),  
 TaxRate Numeric(4, 2) DEFAULT 0,  
 CHECK (TaxRate > -1 AND Rate > -1)  
);  
  
INSERT INTO ITEMS  
VALUES (1, 'Samsung 14" monitor', 7000, 10.5),  
 (2, 'TVS Gold Keyboard', 1000, 10),  
 (3, 'Segate HDD 20GB', 6500, 12.5),  
 (4, 'PIII processor', 8000, 8),  
 (5, 'Logitech Mouse', 500, 5),  
 (6, 'Creative MMK', 4500, 11.5);  
  
INSERT INTO CUSTOMERS  
VALUES (101, 'Raul', '12-22-29', 'Dwarakanagar', 'Vizag', 'AP', '530016', '453343,634333'),  
 (102, 'Denilson', '43-22-22', 'CBM Compound', 'Vizag', 'AP', '530012', '744545'),  
 (103, 'Mendiator', '45-45-52', 'Abid Nagar', 'Vizag', 'AP', '530016', '567434'),  
 (104, 'Figo', '33-34-56', 'Muralinagar', 'Vizag', 'AP', '530021', '875655,876563,872222'),  
 (105, 'Zidane', '23-22-56', 'LB Colony', 'Vizag', 'AP', '530013', '765533');  
  
  
INSERT INTO ORDERS  
VALUES (1001, STR\_TO\_DATE('15-May-2001', '%d-%b-%Y'), STR\_TO\_DATE('10-jun-2001', '%d-%b-%Y'), 102, '43-22-22',  
 'CBM Compound', 'Vizag', 'AP', '530012', '744545'),  
 (1002, STR\_TO\_DATE('15-May-2001', '%d-%b-%Y'), STR\_TO\_DATE('5-jun-2001', '%d-%b-%Y'), 101, '12-22-29',  
 'Dwarakanagar', 'Vizag', 'AP', '530016', '453343,634333'),  
 (1003, STR\_TO\_DATE('17-May-2001', '%d-%b-%Y'), STR\_TO\_DATE('7-jun-2001', '%d-%b-%Y'), 101, '12-22-29',  
 'Dwarakanagar', 'Vizag', 'AP', '530016', '453343,634333'),  
 (1004, STR\_TO\_DATE('18-May-2001', '%d-%b-%Y'), STR\_TO\_DATE('17-jun-2001', '%d-%b-%Y'), 103, '45-45-52',  
 'Abid Nagar', 'Vizag', 'AP', '530016', '567434'),  
 (1005, STR\_TO\_DATE('20-May-2001', '%d-%b-%Y'), STR\_TO\_DATE('3-jun-2001', '%d-%b-%Y'), 104, '33-34-56',  
 'Muralinagar', 'Vizag', 'AP', '530021',  
 '875655,876563,872222'),  
 (1006, STR\_TO\_DATE('23-May-2001', '%d-%b-%Y'), STR\_TO\_DATE('11-jun-2001', '%d-%b-%Y'), 104, '54-22-12',  
 'MVP Colony', 'Vizag', 'AP', '530024', null);  
  
  
INSERT INTO LINEITEMS  
VALUES (1001, 2, 3, 1000, 10.0),  
 (1001, 1, 3, 7000, 15.0),  
 (1001, 4, 2, 8000, 10.0),  
 (1001, 6, 1, 4500, 10.0),  
 (1002, 6, 4, 4500, 20.0),  
 (1002, 4, 2, 8000, 15.0),  
 (1002, 5, 2, 600, 10.0),  
 (1003, 5, 10, 500, 0.0),  
 (1003, 6, 2, 4750, 5.0),  
 (1004, 1, 1, 7000, 10.0),  
 (1004, 3, 2, 6500, 10.0),  
 (1004, 4, 1, 8000, 20.0),  
 (1005, 6, 1, 4600, 10.0),  
 (1005, 2, 2, 900, 10.0),  
 (1006, 2, 10, 950, 20.0),  
 (1006, 4, 5, 7800, 10.0),  
 (1006, 3, 5, 6600, 15.0);  
  
  
  
# Selections  
  
# QN 1  
SELECT \*  
FROM ITEMS  
WHERE ItemName REGEXP 'o.\*o';  
  
# QN 2  
SELECT ItemNo, ItemName, Rate, ROUND(Rate + Rate \* TaxRate / 100) AS 'SELLING PRICE'  
FROM ITEMS;  
  
# QN 3  
SELECT ItemNo, RPAD(UPPER(ItemName), 20, '.'), Rate, TaxRate  
FROM ITEMS;  
  
# QN 4  
SELECT OrdDate, ShipDate  
FROM ORDERS  
WHERE DAYNAME(ShipDate) = 'Monday'  
 AND DATEDIFF(ShipDate, OrdDate) >= 7;  
  
# QN 5  
SELECT \*  
FROM ORDERS  
WHERE MONTH(OrdDate) = MONTH(CURRENT\_DATE)  
 AND YEAR(OrdDate) = YEAR(CURRENT\_DATE);  
  
# QN 6  
SELECT \*  
FROM ORDERS  
WHERE OrdDate BETWEEN DATE\_SUB(LAST\_DAY(DATE\_SUB(CURRENT\_DATE, INTERVAL 1 MONTH)), INTERVAL 7 DAY)  
 AND LAST\_DAY(DATE\_SUB(CURRENT\_DATE, INTERVAL 1 MONTH));  
  
# QN 7  
SELECT OrdNo, DATE\_FORMAT(OrdDate, '%d-%m %H:%i'), COALESCE(ShipDate, DATE\_ADD(OrdDate, INTERVAL 15 DAY))  
FROM ORDERS;  
  
# QN 8  
SELECT COUNT(\*)  
FROM ORDERS;  
  
# QN 9  
SELECT OrdNo, COUNT(\*), AVG(price)  
FROM LINEITEMS  
GROUP BY OrdNo;  
  
# QN 10  
SELECT OrdNo  
FROM LINEITEMS  
GROUP BY OrdNo  
HAVING MAX(Price) > 5000  
 AND SUM(Qty) > 10;  
  
# QN 11  
SELECT MONTHNAME(OrdDate) AS "Month", COUNT(\*) AS 'Order Count'  
FROM ORDERS  
GROUP BY MONTHNAME(OrdDate);  
  
# QN 12  
SELECT CustNo  
FROM ORDERS  
WHERE DATEDIFF(CURRENT\_DATE, OrdDate) < 90  
GROUP BY CustNo  
HAVING COUNT(\*) > 2;  
  
# QN 13  
SELECT Custno, COUNT(\*), MAX(OrdDate)  
FROM ORDERS  
GROUP BY Custno, OrdNo;  
  
# QN 14  
SELECT CustNo, MIN(OrdDate), DATEDIFF(MAX(OrdDate), MIN(OrdDate))  
FROM ORDERS  
GROUP BY CustNo;  
  
# QN 15  
SELECT Ordno, MAX(Price)  
FROM LINEITEMS  
GROUP BY Ordno  
HAVING SUM(Qty \* Price) > 10000;  
  
# QN 16  
SELECT ItemNo, SUM(Qty), MAX(Price), MIN(Price)  
FROM LINEITEMS  
GROUP BY ItemNo;  
  
# QN 17  
SELECT CustNo, DATE(OrdDate), COUNT(DISTINCT OrdDate)  
FROM ORDERS  
GROUP BY CustNo, DATE(OrdDate);  
  
# QN 18  
SELECT OrdNo, CustName, OrdDate, TIMESTAMPDIFF(DAY, ShipDate, OrdDate)  
FROM ORDERS  
 INNER JOIN CUSTOMERS USING (CustNo)  
WHERE ShipDate IS NOT NULL;  
  
# QN 19  
SELECT OrdNo, OrdDate, CustNo, CustName  
FROM ORDERS  
 INNER JOIN CUSTOMERS USING (CustNo)  
WHERE OrdNo IN (SELECT OrdNo FROM LINEITEMS WHERE ItemNo = 5);  
  
# QN 20  
SELECT ItemNo, ItemName, OrdNo, CustName, Qty \* Price AS 'AMOUNT'  
FROM CUSTOMERS  
 INNER JOIN ORDERS USING (CustNo)  
 INNER JOIN LINEITEMS USING (OrdNo)  
 INNER JOIN ITEMS USING (ItemNo);  
  
# QN 21  
SELECT \*  
FROM ORDERS  
WHERE State = 'AP'  
 AND DAYOFWEEK(OrdDate) = 2;  
  
# QN 22  
SELECT \*  
FROM CUSTOMERS  
WHERE CustNo IN (SELECT CustNo  
 FROM ORDERS  
 WHERE OrdNo IN (SELECT OrdNo FROM LINEITEMS GROUP BY OrdNo HAVING SUM(Qty \* Price) > 30000));  
  
# QN 23  
SELECT \*  
FROM ITEMS  
WHERE ItemNo IN (SELECT DISTINCT ItemNo  
 FROM LINEITEMS  
 WHERE OrdNo IN (SELECT OrdNo  
 FROM ORDERS  
 WHERE MONTH(OrdDate) = MONTH(CURRENT\_DATE)  
 AND YEAR(OrdDate) = YEAR(CURRENT\_DATE)));  
  
# QN 24  
SELECT \*  
FROM ORDERS  
WHERE OrdNo IN  
 (SELECT OrdNo FROM LINEITEMS WHERE Price = (SELECT MAX(Price) FROM LINEITEMS WHERE ItemNo = 3) AND ItemNo = 3);  
  
# QN 25  
SELECT \*  
FROM ITEMS  
WHERE ItemNo IN  
 (SELECT ItemNo  
 FROM LINEITEMS  
 GROUP BY ItemNo  
 HAVING SUM(Qty) > 10)  
 OR ItemNo IN  
 (SELECT DISTINCT ItemNo  
 FROM LINEITEMS  
 WHERE OrdNo IN (SELECT OrdNo FROM ORDERS WHERE TIMESTAMPDIFF(DAY, OrdDate, CURRENT\_DATE) < 7));  
  
# QN 26  
SELECT \*  
FROM LINEITEMS  
WHERE Price > (Select AVG(Price) FROM LINEITEMS);  
  
# QN 27  
SELECT \*  
FROM CUSTOMERS  
WHERE CustNo IN (SELECT CustNo  
 FROM ORDERS  
 GROUP BY CustNo  
 HAVING COUNT(CustNo) =  
 (SELECT COUNT(CustNo) FROM ORDERS GROUP BY CustNo ORDER BY COUNT(CustNo) DESC LIMIT 1));  
  
# QN 28  
SELECT \*  
FROM ORDERS  
WHERE OrdNo IN (SELECT OrdNo  
 FROM LINEITEMS  
 INNER JOIN ITEMS USING (ItemNo)  
 WHERE Price > Rate);  
  
# QN 29  
SELECT \*  
FROM CUSTOMERS  
WHERE CustNo IN  
 (SELECT CustNo  
 FROM ORDERS  
 WHERE TIMESTAMPDIFF(DAY, OrdDate, CURRENT\_DATE) >= 15);  
  
# QN 30  
SELECT \*  
FROM ITEMS  
WHERE ItemNo NOT IN (  
 SELECT ItemNo  
 FROM LINEITEMS  
 WHERE OrdNo IN (SELECT OrdNo  
 FROM ORDERS  
 WHERE TIMESTAMPDIFF(MONTH, OrdDate, CURRENT\_DATE) = 1));  
  
# QN 31  
SELECT \*  
FROM ORDERS  
WHERE (MONTH(OrdDate) = MONTH(CURRENT\_DATE) AND YEAR(OrdDate) = YEAR(CURRENT\_DATE))  
 OR OrdNo > 1004;  
  
  
# QN 32  
SELECT \*  
FROM ITEMS  
WHERE ItemNo IN (SELECT DISTINCT ItemNo FROM LINEITEMS WHERE OrdNo IN (SELECT OrdNo FROM ORDERS WHERE CustNo = 102));  
  
# QN 33  
UPDATE ORDERS  
SET ShipDate = (SELECT \* FROM (SELECT MAX(ShipDate) FROM ORDERS) O)  
WHERE OrdNo = 1004;  
  
# QN 34  
SELECT \*  
FROM ITEMS  
WHERE ItemName LIKE '%o%'  
 OR ItemName LIKE '%m%';  
  
# Qn 35  
SELECT \*  
FROM ORDERS  
WHERE MONTH(OrdDate) = 6  
 AND YEAR(OrdDate) = 2000;  
  
# QN 36  
SELECT OrdNo, OrdDate, TIMESTAMPADD(DAY, OrdDate, 15)  
FROM ORDERS  
WHERE ShipDate IS NULL;  
  
# QN 37  
SELECT ItemNo, OrdNo, ROUND(Qty \* Price, -2)  
FROM LINEITEMS  
WHERE Qty > 5  
 OR Price < 5000;  
  
# QN 38  
SELECT ItemNo, ItemName, Rate, Rate \* TaxRate  
FROM ITEMS  
WHERE TaxRate IS NOT NULL;  
  
# QN 39  
SELECT OrdNo, CustNo, OrdDate, DATEDIFF(CURRENT\_DATE, OrdDate), DATE\_ADD(LAST\_DAY(OrdDate), INTERVAL 5 DAY)  
FROM ORDERS;  
  
# QN 40  
SELECT \*  
FROM ORDERS  
WHERE ShipDate IS NOT NULL  
 AND ShipDate < CURRENT\_DATE  
 AND DATEDIFF(CURRENT\_DATE, OrdDate) < 20;  
  
# QN 41  
UPDATE LINEITEMS  
SET Price = Price \* 0.90  
WHERE OrdNo = 1003;  
  
# QN 42  
SELECT \*  
FROM ITEMS  
WHERE LENGTH(ItemName) > 10;  
  
# QN 43  
SELECT \*  
FROM ITEMS  
WHERE POSITION('o' IN ItemName) > 5;  
  
# QN 44  
SELECT SUBSTR(CustName, 1, IF(POSITION(' ' IN CustName) = 0, LENGTH(CustName) - 1, POSITION(' ' IN CustName)))  
FROM CUSTOMERS;  
  
# QN 45  
SELECT ItemNo, UPPER(ItemName)  
FROM ITEMS  
WHERE UPPER(ItemName) LIKE '%m%';  
  
# QN 46  
SELECT \*  
FROM ORDERS  
WHERE MONTH(OrdDate) = MONTH(CURRENT\_DATE)  
 AND YEAR(OrdDate) = YEAR(CURRENT\_DATE);  
  
# QN 47  
# OrdDate cant store a time.  
INSERT INTO ORDERS VALUE (1010, STR\_TO\_DATE('13-Jul-2001', '%d-%b-%Y'), NULL, 105, NULL, NULL, NULL, NULL, NULL, NULL);  
  
# QN 48  
SELECT OrdNo, OrdDate, DATEDIFF(CURRENT\_DATE, COALESCE(ShipDate, CURRENT\_DATE))  
FROM ORDERS;  
  
# QN 49  
SELECT ItemNo, Price, Qty, IF(ItemNo = 1, 10, IF(ItemNo = 6, 7, 8))  
FROM LINEITEMS;  
  
# QN 50  
SELECT SUM(Qty \* Price), COUNT(OrdNo)  
FROM LINEITEMS;  
  
# QN 51  
# Assuming that the number of orders refers to the current month than current year as that doesnt make sense  
SELECT CustNo, MONTHNAME(OrdDate), COUNT(\*)  
FROM ORDERS  
GROUP BY CustNo, MONTHNAME(OrdDate);  
  
# QN 52  
SELECT ItemNo, MAX(Price) - MIN(Price)  
FROM LINEITEMS  
GROUP BY ItemNo;  
  
# QN 53  
SELECT COUNT(\*)  
FROM ORDERS  
WHERE ShipDate IS NULL;  
  
# QN 54  
SELECT OrdNo, AVG(Price)  
FROM ORDERS  
 INNER JOIN LINEITEMS USING (OrdNo)  
WHERE DATEDIFF(CURRENT\_DATE, OrdDate) <= 15  
GROUP BY OrdNo;  
  
# QN 55  
SELECT YEAR(OrdDate), COUNT(\*)  
FROM ORDERS  
WHERE DATEDIFF(ShipDate, OrdDate) <= 10  
GROUP BY YEAR(OrdDate);  
  
# QN 56  
SELECT State, COUNT(\*)  
FROM CUSTOMERS  
WHERE CustName LIKE '%nike%'  
GROUP BY State;  
  
# QN 57  
SELECT CustName  
FROM CUSTOMERS  
WHERE CustNo IN (  
 SELECT CustNo  
 FROM ORDERS  
 GROUP BY CustNo, MONTH(OrdDate), YEAR(OrdDate)  
 HAVING COUNT(\*) > 2);  
  
# QN 58  
SELECT COUNT(\*)  
FROM ORDERS  
GROUP BY CustNo  
ORDER BY COUNT(\*) DESC  
LIMIT 1;  
  
# QN 59  
SELECT CustNo, SUM(IF(ShipDate IS NOT NULL, 1, 0)), SUM(IF(ShipDate IS NOT NULL, 0, 1))  
FROM ORDERS  
GROUP BY CustNo;  
  
# QN 60  
SELECT OrdNo, ItemNo, ItemName, Price, Rate  
FROM LINEITEMS  
 INNER JOIN ITEMS USING (ItemNo);  
  
# QN 61  
SELECT OrdNo, ItemNo, ItemName, Price, Rate  
FROM LINEITEMS  
 INNER JOIN ITEMS USING (ItemNo)  
WHERE Price > Rate;  
  
# QN 62  
SELECT ItemName, ItemNo, OrdNo, Rate - Price  
FROM LINEITEMS  
 INNER JOIN ITEMS USING (ItemNo)  
WHERE Price != Rate;  
  
# QN 63  
SELECT C.CustNo, C.CustName, O.OrdNo, O.OrdDate  
FROM CUSTOMERS C  
 INNER JOIN ORDERS O  
WHERE O.Address1 = C.Address1  
 AND O.Address2 = C.Address2  
 AND O.City = C.City  
 AND O.State = C.State  
 AND O.PinCode = C.PinCode;  
  
# QN 64  
SELECT ItemNo, ItemName, OrdNo, Qty  
FROM LINEITEMS  
 INNER JOIN ITEMS USING (ItemNo);  
  
# QN 65  
SELECT C.CustNo, COUNT(\*)  
FROM CUSTOMERS C  
 INNER JOIN ORDERS O  
WHERE O.CustNo = C.CustNo  
 AND C.City = 'VIZAG'  
GROUP BY C.CustNo;  
  
# QN 66  
SELECT OrdNo, CustName, DATEDIFF(CURRENT\_DATE, OrdDate)  
FROM ORDERS  
 INNER JOIN CUSTOMERS USING (CustNo)  
WHERE DATEDIFF(CURRENT\_DATE, OrdDate) < 10  
 AND ShipDate IS NULL;  
  
# QN 67  
SELECT CustName, SUM(Qty \* Price)  
FROM CUSTOMERS  
 INNER JOIN ORDERS USING (CustNo)  
 INNER JOIN LINEITEMS USING (OrdNo)  
GROUP BY CustNo;  
  
# QN 68  
SELECT \*  
FROM ITEMS  
WHERE Rate = (SELECT MAX(Rate) FROM ITEMS);  
  
# QN 69  
SELECT \*  
FROM CUSTOMERS  
WHERE CustNo IN (SELECT CustNo FROM ORDERS GROUP BY CustNo HAVING COUNT(\*) > 5);  
  
# QN 70  
SELECT \*  
FROM CUSTOMERS  
WHERE CustNo NOT IN (SELECT DISTINCT CustNo FROM ORDERS);  
  
# QN 71  
SELECT \*  
FROM CUSTOMERS  
WHERE CustNo IN (SELECT DISTINCT CustNo FROM ORDERS WHERE TIMESTAMPDIFF(MONTH, OrdDate, CURRENT\_DATE) <= 6);  
  
# QN 72  
SELECT \*  
FROM ITEMS  
WHERE ItemNo IN (SELECT ItemNo FROM LINEITEMS WHERE Price > 5000 GROUP BY ItemNo HAVING SUM(Qty) > 50);  
  
# QN 73  
SELECT \*  
FROM ORDERS  
WHERE OrdNo IN (SELECT OrdNo FROM LINEITEMS GROUP BY OrdNo HAVING COUNT(\*) > 5)  
 OR CustNo IN (SELECT CustNo FROM CUSTOMERS WHERE CUSTOMERS.Phone LIKE '541%');  
  
# QN 74  
UPDATE ITEMS  
SET Rate = (SELECT \* FROM (SELECT MAX(Rate) FROM ITEMS) R)  
WHERE ItemNo = 1;  
  
# QN 75  
DELETE  
FROM CUSTOMERS  
WHERE CustNo NOT IN (SELECT DISTINCT CustNo FROM ORDERS);  
  
# QN 76  
ALTER TABLE ITEMS  
 CHANGE Rate Price Numeric(8, 2);  
  
# Changed Items.Price back to Items.Rate  
ALTER TABLE ITEMS  
 CHANGE Price Rate Numeric(8, 2);  
  
# QN 77 - Same as 27  
  
# QN 78  
SELECT \*  
FROM CUSTOMERS  
WHERE CustNo NOT IN (SELECT DISTINCT CustNo  
 FROM ORDERS  
 WHERE TIMESTAMPDIFF(MONTH, OrdDate, CURRENT\_DATE) = 0);  
  
# QN 79  
SELECT \*  
FROM ITEMS  
WHERE ItemNo NOT IN (  
 SELECT ItemNo  
 FROM LINEITEMS  
 WHERE OrdNo IN (SELECT OrdNo  
 FROM ORDERS  
 WHERE TIMESTAMPDIFF(MONTH, OrdDate, CURRENT\_DATE) = 1)  
 AND OrdNo NOT IN (SELECT OrdNo  
 FROM ORDERS  
 WHERE TIMESTAMPDIFF(MONTH, OrdDate, CURRENT\_DATE) = 0));  
  
# QN 80  
SELECT \*  
FROM ITEMS  
WHERE ItemNo IN (Select DISTINCT ItemNo  
 FROM LINEITEMS  
 WHERE OrdNo IN (SELECT OrdNo  
 FROM ORDERS  
 WHERE CustNo IN (SELECT CustNo FROM ORDERS GROUP BY CustNo HAVING COUNT(\*) > 3)));  
  
# QN 81  
SELECT \*  
FROM ORDERS  
WHERE ShipDate IS NOT NULL  
 AND TIMESTAMPDIFF(DAY, OrdDate, ShipDate) > (SELECT AVG(TIMESTAMPDIFF(DAY, OrdDate, ShipDate)) FROM ORDERS);  
  
# QN 82  
SELECT \*  
FROM ITEMS  
WHERE Rate > (SELECT MAX(Price) FROM LINEITEMS WHERE ITEMS.ItemNo = LINEITEMS.ItemNo);  
  
# QN 84  
SELECT \*  
FROM ITEMS  
WHERE ItemNo IN  
 (SELECT DISTINCT Rate FROM (SELECT DISTINCT Rate FROM ITEMS ORDER BY Rate LIMIT 2) T ORDER BY Rate DESC LIMIT 1);  
  
# QN 85  
CREATE TABLE 'COMPORDERS'  
(  
 OrdNo numeric(5) REFERENCES ORDERS (OrdNo),  
 CustName varchar(20) NOT NULL,  
 OrdDate date,  
 ShipDate date,  
 OrderShipDiff TIMESTAMP  
);  
  
  
  
  
# 1  
DELIMITER //  
CREATE PROCEDURE *UpdateListItems*()  
BEGIN  
 DECLARE order\_no NUMERIC(5);  
 DECLARE current\_rate NUMERIC(8, 2);  
  
 SELECT MAX(OrdNo)  
 INTO order\_no  
 FROM ORDERS  
 WHERE CustNo = 106;  
 SELECT Rate  
 INTO current\_rate  
 FROM ITEMS  
 WHERE ItemNo = 3;  
 INSERT INTO LINEITEMS  
 VALUES (order\_no, 3, 2, current\_rate, 8);  
END //  
  
DELIMITER ;  
  
# 2  
DELIMITER //  
  
CREATE PROCEDURE *ChangeRate*()  
BEGIN  
 DECLARE avg\_rate NUMERIC(8, 2);  
 DECLARE current\_rate NUMERIC(8, 2);  
 SELECT AVG(Price) INTO avg\_rate FROM LINEITEMS WHERE ItemNo = 5;  
 SELECT Rate INTO current\_rate FROM ITEMS WHERE ItemNo = 5;  
 UPDATE ITEMS SET Rate = GREATEST(avg\_rate, current\_rate) WHERE ItemNo = 5;  
end //  
  
DELIMITER ;  
  
# 3  
DELIMITER //  
CREATE PROCEDURE *InsertLineItems*()  
BEGIN  
 DECLARE order\_no NUMERIC(5);  
 DECLARE item\_no NUMERIC(5);  
 DECLARE item\_rate, current\_rate NUMERIC(8, 2);  
 DECLARE item\_discount NUMERIC(4, 2);  
 SELECT MAX(OrdNo) INTO order\_no FROM ORDERS WHERE CustNo = 102;  
 SELECT ItemNo, Rate INTO item\_no, current\_rate FROM ITEMS WHERE UPPER(ItemName) = 'PIII PROCESSOR';  
 SELECT MIN(price) INTO item\_rate FROM LINEITEMS WHERE ItemNo = item\_no;  
 IF current\_rate > item\_rate THEN  
 SET item\_discount = 10;  
 ELSE  
 SET item\_discount = 0;  
 END IF;  
 INSERT INTO LINEITEMS VALUE (order\_no, item\_no, 2, item\_rate, item\_discount);  
END //  
DELIMITER ;  
  
# 4  
DELIMITER //  
CREATE PROCEDURE *MaxMissingOrderNo*()  
BEGIN  
 DECLARE min\_order\_no, max\_order\_no NUMERIC(5);  
 DECLARE x NUMERIC(5);  
 DECLARE cnt NUMERIC(2);  
 SELECT MIN(OrdNo), MAX(OrdNo) INTO min\_order\_no, max\_order\_no FROM ORDERS;  
 SET x = max\_order\_no;  
 loop\_label:  
 LOOP  
 SELECT COUNT(\*) INTO cnt FROM ORDERS WHERE OrdNo = x;  
 IF cnt = 0 THEN  
 SELECT x;  
 LEAVE loop\_label;  
 ELSE  
 SET x = x - 1;  
 ITERATE loop\_label;  
 END IF;  
 END LOOP;  
END //  
DELIMITER ;  
  
# 5  
SELECT CustName  
FROM CUSTOMERS  
WHERE CustNo IN (SELECT CustNo  
 FROM ORDERS  
 WHERE OrdNo IN (SELECT OrdNo FROM LINEITEMS WHERE Qty \* Price > 10000)  
 GROUP BY CustNo  
 HAVING COUNT(\*) > 3);  
  
# 6  
DELIMITER //  
CREATE PROCEDURE *Change\_Rate*()  
BEGIN  
 DECLARE finished INTEGER DEFAULT 0;  
 DECLARE item\_no NUMERIC(5);  
 DECLARE avg\_rate, current\_rate NUMERIC(8, 2);  
 DECLARE item\_count NUMERIC(2);  
 DECLARE itemCursor CURSOR FOR SELECT ItemNo FROM ITEMS;  
 DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;  
  
 open itemCursor;  
  
 getItem:  
 LOOP  
 FETCH itemCursor INTO item\_no;  
 IF finished = 1 THEN  
 LEAVE getItem;  
 END IF;  
  
 SELECT COUNT(\*) INTO item\_count FROM LINEITEMS WHERE ItemNo = item\_no;  
  
 SELECT AVG(Price) INTO avg\_rate FROM LINEITEMS WHERE ItemNo = 5;  
 SELECT Rate INTO current\_rate FROM ITEMS WHERE ItemNo = item\_no;  
  
 IF item\_count > 5 THEN  
 UPDATE ITEMS SET Rate = Rate \* 1.1;  
  
 ELSEIF avg\_rate > current\_rate THEN  
 UPDATE ITEMS SET Rate = Rate \* 1.02;  
  
 ELSE  
 UPDATE ITEMS SET Rate = Rate \* 0.97;  
  
 END IF;  
  
 END LOOP getItem;  
 CLOSE itemCursor;  
  
END//  
DELIMITER ;  
  
# 7  
  
CREATE TABLE CUSTSUM  
(  
 CustNo NUMERIC(5),  
 CustName VARCHAR(20),  
 NoOfOrders NUMERIC(2),  
 RecentOrder DATE,  
 TotalAmount NUMERIC(8, 2)  
);  
  
DELIMITER //  
CREATE PROCEDURE *CreateCustSum*()  
BEGIN  
 DECLARE finished INTEGER DEFAULT 0;  
 DECLARE customer\_no NUMERIC(5);  
 DECLARE customer\_name VARCHAR(20);  
 DECLARE no\_of\_orders\_placed NUMERIC(2);  
 DECLARE recent\_order\_date DATE;  
 DECLARE total\_amount NUMERIC(8, 2);  
 DECLARE customer\_cursor CURSOR FOR SELECT CustNo, CustName FROM CUSTOMERS;  
  
 DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;  
  
 OPEN customer\_cursor;  
  
 getCustomer:  
 LOOP  
 FETCH customer\_cursor INTO customer\_no, customer\_name;  
 IF finished = 1 THEN  
 LEAVE getCustomer;  
 END IF;  
 SELECT COUNT(\*), MAX(OrdDate), SUM(Qty \* Price)  
 INTO no\_of\_orders\_placed, recent\_order\_date, total\_amount  
 FROM ORDERS  
 NATURAL JOIN LINEITEMS  
 WHERE CustNo = customer\_no;  
  
 INSERT INTO CUSTSUM  
 VALUES (customer\_no, customer\_name, no\_of\_orders\_placed, recent\_order\_date, total\_amount);  
 END LOOP getCustomer;  
 CLOSE customer\_cursor;  
  
END //  
DELIMITER ;  
  
# 8  
SELECT ItemName  
FROM ITEMS  
WHERE ItemNo IN  
 (SELECT DISTINCT ItemNo FROM ITEMS I WHERE Rate < (SELECT AVG(Price) FROM LINEITEMS WHERE I.ItemNo = ItemNo))  
 OR ItemNo IN  
 (SELECT DISTINCT ItemNo FROM LINEITEMS GROUP BY ItemNo HAVING SUM(Qty) < 10);  
  
# 9  
DELIMITER //  
CREATE PROCEDURE *PS8\_9*(IN order\_no NUMERIC(5),  
 IN item\_no NUMERIC(5))  
BEGIN  
 DECLARE item\_price NUMERIC(8, 2);  
 SELECT Rate INTO item\_price FROM ITEMS WHERE ItemNo = item\_no;  
 INSERT INTO LINEITEMS  
 VALUES (order\_no, item\_no, 1, item\_price, 10);  
END//  
DELIMITER ;  
  
# 10  
DELIMITER //  
CREATE FUNCTION *MissingOrderNo*()  
 RETURNS NUMERIC(5)  
 DETERMINISTIC  
BEGIN  
 DECLARE order\_count NUMERIC(2);  
 DECLARE min\_order\_no NUMERIC(5);  
 DECLARE max\_order\_no NUMERIC(5);  
 DECLARE x NUMERIC(5);  
  
 SELECT MIN(OrdNo), MAX(OrdNo) INTO min\_order\_no, max\_order\_no FROM ORDERS;  
  
 SET x = min\_order\_no;  
  
 getOrder:  
 LOOP  
 IF x > max\_order\_no THEN  
 LEAVE getOrder;  
 END IF;  
 SELECT COUNT(\*) INTO order\_count FROM ORDERS WHERE OrdNo = x;  
 IF order\_count = 0 THEN  
 RETURN (x);  
 END IF;  
 SET x = x + 1;  
 END LOOP getOrder;  
 RETURN NULL;  
END //  
DELIMITER ;  
  
# 11  
DELIMITER //  
CREATE FUNCTION *ReturnCustomerName*(  
 order\_no NUMERIC(5)  
)  
 RETURNS VARCHAR(20)  
 DETERMINISTIC  
BEGIN  
 DECLARE customer\_name VARCHAR(20);  
 SELECT CustName  
 INTO customer\_name  
 FROM CUSTOMERS  
 WHERE CustNo = (SELECT DISTINCT CustNo FROM ORDERS WHERE OrdNo = order\_no);  
 RETURN (customer\_name);  
END //  
DELIMITER ;  
  
# 12  
DELIMITER //  
CREATE PROCEDURE *InsertIntoLineItems*(IN item\_no NUMERIC(5),  
 IN price NUMERIC(8, 2),  
 IN quantity NUMERIC(3))  
BEGIN  
 DECLARE order\_no NUMERIC(5);  
 DECLARE current\_rate NUMERIC(8, 2);  
 DECLARE item\_count NUMERIC(3);  
 DECLARE total\_amount NUMERIC(8, 2);  
  
 SELECT MAX(OrdNo) INTO order\_no FROM ORDERS;  
  
 SELECT Rate INTO current\_rate FROM ITEMS WHERE ItemNo = item\_no;  
  
 IF price > current\_rate THEN  
 SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Price is greater than current rate';  
 END IF;  
  
 SELECT COUNT(\*) INTO item\_count FROM LINEITEMS WHERE OrdNo = order\_no AND ItemNo = item\_no;  
  
 IF item\_count != 0 THEN  
 SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Item already exists in the order';  
 END IF;  
  
 SELECT SUM(Qty \* Price) INTO total\_amount FROM LINEITEMS WHERE OrdNo = order\_no;  
  
 IF (quantity \* price) + total\_amount > 50000 THEN  
 SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Total amount exceeds 50000';  
 END IF;  
  
 INSERT INTO LINEITEMS  
 VALUES (order\_no, item\_no, quantity, price, 0);  
  
END //  
DELIMITER ;  
  
# 13  
DELIMITER //  
CREATE TRIGGER CheckItemCount  
 BEFORE INSERT  
 ON LINEITEMS  
 FOR EACH ROW  
BEGIN  
 DECLARE row\_count INT;  
 SELECT COUNT(\*) INTO row\_count FROM LINEITEMS;  
  
 IF row\_count > 5 THEN  
 SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Number of items per order is greater than 5';  
 END IF;  
  
END //  
DELIMITER ;  
  
# 14  
DELIMITER //  
CREATE TRIGGER CheckTime  
 BEFORE UPDATE  
 ON ITEMS  
 FOR EACH ROW  
BEGIN  
 IF CURRENT\_TIME() < STR\_TO\_DATE('9 am', '%I %p') OR CURRENT\_TIME() > STR\_TO\_DATE('9 pm', '%I %p') THEN  
 SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Changes cannot be made before 9 am and after 9 pm';  
 END IF;  
  
END //  
DELIMITER ;  
  
# 15  
DELIMITER //  
CREATE TRIGGER CheckRateChange  
 BEFORE UPDATE  
 ON ITEMS  
 FOR EACH ROW  
BEGIN  
 DECLARE diff\_rate NUMERIC(4, 2);  
 SET diff\_rate = NEW.Rate - OLD.Rate;  
 IF diff\_rate > OLD.Rate \* 0.25 THEN  
 SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Invalid rate amount. Change is too big';  
 END IF;  
END //  
DELIMITER ;  
  
# 16  
DELIMITER //  
CREATE PROCEDURE *PS8\_16*()  
BEGIN  
 DECLARE order\_no NUMERIC(5);  
 DECLARE order\_date DATE;  
 DECLARE customer\_no NUMERIC(5);  
 DECLARE ship\_date DATE;  
 DECLARE address1 VARCHAR(50);  
 DECLARE address2 VARCHAR(50);  
 DECLARE city VARCHAR(30);  
 DECLARE state VARCHAR(30);  
 DECLARE pincode VARCHAR(10);  
 DECLARE phone VARCHAR(30);  
 DECLARE price NUMERIC(8, 2);  
 SELECT MAX(OrdNo) + 1  
 INTO order\_no  
 FROM ORDERS;  
 SET order\_date = SUBDATE(CURRENT\_DATE, 1);  
 SET customer\_no = 103;  
 SET ship\_date = ADDDATE(order\_date, 15);  
 SELECT Address1, Address2, City, State, Pincode, Phone  
 INTO address1, address2, city, state, pincode, phone  
 FROM CUSTOMERS;  
  
 INSERT INTO ORDERS  
 VALUES (order\_no, @order\_date, ship\_date, customer\_no, address1, address2, city, state, pincode, phone);  
  
 SELECT MIN(Price) INTO price FROM LINEITEMS WHERE ItemNo = 4;  
  
 INSERT INTO LINEITEMS  
 VALUES (order\_no, 4, 2, price, 0);  
  
END//  
DELIMITER ;  
  
# 17  
DELIMITER //  
CREATE PROCEDURE *DisplayItems*()  
BEGIN  
 DECLARE finished INTEGER DEFAULT 0;  
 DECLARE item\_name VARCHAR(20);  
 DECLARE item\_no NUMERIC(5);  
 DECLARE item\_count NUMERIC(2);  
 DECLARE item\_cursor CURSOR FOR SELECT ItemNo, SUM(Qty) FROM LINEITEMS GROUP BY ItemNo;  
 DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;  
  
 OPEN item\_cursor;  
 item\_loop:  
 LOOP  
 FETCH item\_cursor INTO item\_no, item\_count;  
 IF finished = 1 THEN  
 LEAVE item\_loop;  
 END IF;  
 IF item\_count != 0 THEN  
 SELECT ItemName INTO item\_name FROM ITEMS WHERE ItemNo = item\_no;  
 SELECT item\_name AS ItemName, item\_count AS ItemCount;  
 END IF;  
 END LOOP;  
  
 CLOSE item\_cursor;  
END //  
DELIMITER ;  
  
# 18  
DELIMITER //  
CREATE PROCEDURE *PS8\_18*()  
BEGIN  
 DECLARE item\_rate NUMERIC(8, 2);  
  
 SELECT Rate INTO item\_rate FROM ITEMS WHERE ItemNo = 4;  
  
 INSERT INTO LINEITEMS  
 VALUES (1003, 4, 1, item\_rate, 5);  
  
END //  
DELIMITER ;  
  
# 19  
DELIMITER //  
CREATE PROCEDURE *PS8\_19*()  
BEGIN  
 DECLARE item\_rate NUMERIC(8, 2);  
 DECLARE item\_count NUMERIC(2);  
 DECLARE total\_amount NUMERIC(8, 2);  
 DECLARE order\_date DATE;  
  
 SELECT COUNT(\*) INTO item\_count FROM LINEITEMS WHERE ItemNo = 4 AND OrdNo = 1003;  
 IF item\_count >= 1 THEN  
 SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Item 4 alredy exists for order 1003';  
 END IF;  
  
 SELECT SUM(Qty \* Price) INTO total\_amount FROM LINEITEMS WHERE OrdNo = 1003;  
 IF total\_amount > 30000 THEN  
 SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Total amount of order is crossing 30000';  
 END IF;  
  
 SELECT OrdDate INTO order\_date FROM ORDERS WHERE OrdNo = 1003;  
 IF DATEDIFF(CURRENT\_DATE, order\_date) > 4 THEN  
 SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Order was placed before 4 days';  
 END IF;  
  
 SELECT Rate INTO item\_rate FROM ITEMS WHERE ItemNo = 4;  
  
 INSERT INTO LINEITEMS  
 VALUES (1003, 4, 1, item\_rate, 5);  
  
END //  
DELIMITER ;  
  
# 20  
DELIMITER //  
CREATE PROCEDURE *DisplayAmount*()  
BEGIN  
 DECLARE finished INTEGER DEFAULT 0;  
 DECLARE cust\_name VARCHAR(20);  
 DECLARE cust\_no NUMERIC(5);  
 DECLARE cust\_cursor CURSOR FOR SELECT CustNo, CustName FROM CUSTOMERS LIMIT 5;  
 DECLARE amount NUMERIC(8, 2);  
 DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;  
  
 OPEN cust\_cursor;  
  
 cust\_loop:  
 LOOP  
 FETCH cust\_cursor INTO cust\_no, cust\_name;  
 IF finished = 1 THEN  
 LEAVE cust\_loop;  
 END IF;  
  
 SELECT SUM(Qty \* Price)  
 INTO amount  
 FROM LINEITEMS  
 WHERE OrdNo IN (SELECT OrdNo FROM ORDERS WHERE CustNo = cust\_no);  
  
 SELECT cust\_name AS CustomerName, amount AS TotalAmount;  
 END LOOP;  
  
 CLOSE cust\_cursor;  
  
END //  
DELIMITER ;  
  
# 21  
DELIMITER //  
CREATE PROCEDURE *PS8\_20*()  
BEGIN  
 DECLARE finished INTEGER DEFAULT 0;  
 DECLARE item\_no NUMERIC(5);  
 DECLARE item\_cursor CURSOR FOR SELECT ItemNo, Rate FROM ITEMS;  
 DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;  
 DECLARE item\_count NUMERIC(2);  
 DECLARE item\_qty NUMERIC(2);  
 DECLARE item\_rate NUMERIC(8, 2);  
 OPEN item\_cursor;  
  
 item\_loop:  
 LOOP  
 FETCH item\_cursor INTO item\_no, item\_rate;  
 IF finished = 1 THEN  
 LEAVE item\_loop;  
 END IF;  
  
 SELECT COUNT(\*), SUM(Qty) INTO item\_count, item\_qty FROM LINEITEMS WHERE ItemNo = item\_no;  
  
 IF item\_count > 5 OR item\_qty > 25 THEN  
 UPDATE ITEMS SET Rate = 1.1 \* Rate WHERE ItemNo = item\_no;  
 ELSE  
 SELECT COUNT(\*) INTO item\_count FROM LINEITEMS WHERE ItemNo = item\_no AND Price > item\_rate;  
 IF item\_count > 0 THEN  
 UPDATE ITEMS SET Rate = 1.05 \* Rate WHERE ItemNo = item\_no;  
 END IF;  
 END IF;  
  
 END LOOP;  
  
 CLOSE item\_cursor;  
END //  
DELIMITER ;  
  
# 22  
DELIMITER //  
CREATE TRIGGER PreventIncrease  
 BEFORE UPDATE  
 ON LINEITEMS  
 FOR EACH ROW  
BEGIN  
 IF NEW.Price > OLD.Price THEN;  
 SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Not allowed to change price of items';  
 END IF;  
END //  
DELIMITER ;  
  
# 23  
DELIMITER //  
CREATE TRIGGER CheckTime  
 BEFORE UPDATE  
 ON ORDERS  
 FOR EACH ROW  
BEGIN  
 IF CURRENT\_TIME() < STR\_TO\_DATE('9 am', '%I %p') OR CURRENT\_TIME() > STR\_TO\_DATE('9 pm', '%I %p') THEN  
 SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Changes cannot be made before 9 am and after 9 pm';  
 END IF;  
  
END //  
DELIMITER ;  
  
# 24  
DELIMITER //  
CREATE FUNCTION *NextOrderno*(  
 order\_no NUMERIC(5)  
)  
 RETURNS NUMERIC(5)  
 DETERMINISTIC  
BEGIN  
 DECLARE order\_count NUMERIC(2);  
 DECLARE max\_order\_no NUMERIC(5);  
 DECLARE x NUMERIC(5);  
 SET x = order\_no + 1;  
 getorder:  
 LOOP  
 IF x > max\_order\_no THEN  
 LEAVE getOrder;  
 END IF;  
 SELECT COUNT(\*) INTO order\_count FROM ORDERS WHERE OrdNo = x;  
 IF order\_count != 0 THEN  
 RETURN (x);  
 END IF;  
 SET x = x + 1;  
 END LOOP;  
 RETURN NULL;  
END //  
DELIMITER ;  
  
# 25  
DELIMITER //  
CREATE FUNCTION *MissingOrderNo*()  
 RETURNS NUMERIC(5)  
 DETERMINISTIC  
BEGIN  
 DECLARE order\_count NUMERIC(2);  
 DECLARE min\_order\_no NUMERIC(5);  
 DECLARE max\_order\_no NUMERIC(5);  
 DECLARE x NUMERIC(5);  
  
 SELECT MIN(OrdNo), MAX(OrdNo) INTO min\_order\_no, max\_order\_no FROM ORDERS;  
  
 SET x = min\_order\_no;  
  
 getOrder:  
 LOOP  
 IF x > max\_order\_no THEN  
 LEAVE getOrder;  
 END IF;  
 SELECT COUNT(\*) INTO order\_count FROM ORDERS WHERE OrdNo = x;  
 IF order\_count = 0 THEN  
 RETURN (x);  
 END IF;  
 SET x = x + 1;  
 END LOOP getOrder;  
 RETURN NULL;  
END //  
DELIMITER ;