```
//Database Level

CREATE DATABASE order_db_mgmt;

DROP DATABASE order_db_mgmt;

CREATE DATABASE IF NOT EXISTS order_db_mgmt;

DROP DATABASE IF EXISTS order_db_mgmt;

CREATE DATABASE IF NOT EXISTS order_db_mgmt;

SHOW CREATE DATABASE order_db_mgmt \G

SHOW DATABASES;

Use order_db_mgmt;

SELECT DATABASE();


//Table level

SHOW TABLES;

CREATE TABLE IF NOT EXISTS customer (
customerID INT UNSIGNED NOT NULL AUTO_INCREMENT,
customername CHAR(30) NOT NULL DEFAULT '',
customeraddress VARCHAR(30) NOT NULL DEFAULT '',
customermobno INT UNSIGNED NOT NULL,
PRIMARY KEY (customerID)
);

DESCRIBE customer;

SHOW CREATE TABLE customer;

SHOW CREATE TABLE customer \G

CREATE TABLE IF NOT EXISTS products (
productID INT UNSIGNED NOT NULL AUTO_INCREMENT primary
key,
productCode CHAR(3) NOT NULL DEFAULT '',
name VARCHAR(30) NOT NULL DEFAULT '',
```

```
quantity INT UNSIGNED NOT NULL DEFAULT 0,
price DECIMAL(7,2) NOT NULL DEFAULT 99999.99);

ALTER TABLE products ADD COLUMN supplierID INT UNSIGNED
NOT NULL;

Desc products;

ALTER TABLE products DROP supplierID;
```

//Row Level

```
INSERT INTO products VALUES (1001, 'PEN', 'Pen Red', 5000, 1.23);

INSERT INTO products VALUES
(NULL, 'PEN', 'Pen Blue', 8000, 1.25),
(NULL, 'PEN', 'Pen Black', 2000, 1.25);
```

//null resulting auto-increment

```
INSERT INTO products (productCode, name, quantity, price) VALUES
('PEC', 'Pencil 2B', 10000, 0.48),
('PEC', 'Pencil 2H', 8000, 0.49);

INSERT INTO products (productCode, name) VALUES ('PEC', 'Pencil HB');

INSERT INTO products values (NULL, NULL, NULL, NULL, NULL);

INSERT INTO products SET productCode=”PEC”, name=”Pencil HB”;

Select * from products;

SELECT productid, name FROM products;

SELECT productid, name FROM products where productCode=”PEC”;
```

```
SELECT * FROM products where productCode=”PEC”;


SELECT 1+1;
Select now();
SELECT 1+1, NOW();
UPDATE products SET supplierID = 501;

UPDATE products SET price = price * 1.1;

SELECT * FROM products;

UPDATE products SET quantity = quantity - 100 WHERE name = 'Pen Red';

SELECT * FROM products WHERE name = 'Pen Red';

UPDATE products SET quantity = quantity + 50, price = 1.23 WHERE name
= 'Pen Red';
SELECT * FROM products WHERE name = 'Pen Red';

DELETE FROM products WHERE productid=1007;

DELETE FROM products;  //very careful before using


CREATE TABLE suppliers (
supplierID INT UNSIGNED NOT NULL AUTO_INCREMENT,
name VARCHAR(30) NOT NULL DEFAULT '',
phone CHAR(8) NOT NULL DEFAULT '',
PRIMARY KEY (supplierID)
);

DESCRIBE suppliers;

INSERT INTO suppliers VALUE
(501, 'ABC Traders', '88881111'),
(502, 'XYZ Company', '88882222'),
(503, 'QQ Corp', '88883333');
```

```
SELECT * FROM suppliers;

ALTER TABLE products ADD FOREIGN KEY (supplierID)
REFERENCES suppliers (supplierID);
```

//String Pattern Matching - LIKE and NOT LIKE

```
SELECT name, price FROM products WHERE name LIKE 'PENCIL%';
//begineeing with

SELECT name, price FROM products WHERE name LIKE 'P__ %';

SELECT * FROM products WHERE quantity >= 5000 AND name LIKE
'Pen %';


SELECT * FROM products WHERE quantity >= 5000 AND price < 1.24
AND name LIKE 'Pen %';


SELECT * FROM products WHERE NOT (quantity >= 5000 AND name
LIKE 'Pen %');


SELECT * FROM products WHERE name IN ('Pen Red', 'Pen Black');

SELECT * FROM products WHERE (price BETWEEN 1.0 AND 2.0) AND
(quantity BETWEEN 1000 AND 2000);

SELECT * FROM products WHERE productCode IS NULL;

SELECT * FROM products ORDER BY price DESC;

SELECT * FROM products WHERE name LIKE 'Pen %' ORDER BY price
DESC, quantity;

SELECT * FROM products ORDER BY price LIMIT 2;
```

SELECT CONCAT(productCode, ' - ', name) AS `Product Description`, price FROM products;

CREATE TABLE IF NOT EXISTS orders (
orderID varchar(30) NOT NULL primary key,
orderdate date,
customerID INT UNSIGNED NOT NULL, foreign key(customerID) references customer(customerID));

insert into customer value(1,"abc","kolhapur",123456789),(2,"def","pune",123789456),(3,"ghi","mumbai",123321123);

insert into orders value("o1","2022-12-12",2);

insert into orders value("o2","2022-1-12",3);

insert into orders value("o3","2023-1-12",3);

create table order_product (orderID varchar(30) NOT NULL, productID INT UNSIGNED NOT NULL, purchase_quantity int default 1, foreign key(orderID) references orders(orderID), foreign key(productID) references products(productID));

SELECT DISTINCT price AS `Distinct Price` FROM products;

SELECT DISTINCT price, name FROM products;

SELECT * FROM products ORDER BY productCode, productid

SELECT * FROM products GROUP BY productCode;

LOAD DATA LOCAL INFILE 'E:/DMSL/products.csv' INTO TABLE products
COLUMNS TERMINATED BY ','
LINES TERMINATED BY '\r\n';

select * from customer c, orders o,order_product op whe
re c.customerID=o.customerID and o.orderID=op.orderID;

select * from customer c, orders o,order_product op where c.customerID=o.customerID and o.orderID=op.orderID \G

select * from customer c, orders o,order_product op whe re c.customerID=o.customerID and o.orderID=op.orderID and c.customername="def" \

select * from customer c, orders o,order_product op whe re c.customerID=o.customerID and o.orderID=op.orderID and c.customername="def";

select customername,count(productID) as "total product pu rchased" from customer c, orders o,order_product op where c.customerID=o.custome rID and o.orderID=op.orderID and c.customername="def";

```
SELECT products.name, price, suppliers.name
FROM products
JOIN suppliers ON products.supplierID = suppliers.supplierID
WHERE price < 0.6;

SELECT products.name, price, suppliers.name
FROM products, suppliers
WHERE products.supplierID = suppliers.supplierID
AND price < 0.6;
```

JOIN

select c.customerID,customername,orderID,o.customerID from customer c inner join orders o where c.customerID=o.customerID;

select c.customerID,customername,orderID,o.customerID from customer c inner join orders o on c.customerID=o.customerID;

select c.customerID,customername,orderID,o.customerID from customer c join orders o where c.customerID=o.customerID and c.customername="ghi";

```
select c.customerID,customername,orderID,o.customerID from customer c
inner join orders o on c.customerID=o.customerID and
c.customername="ghi";

select c.customerID,customername,orderID,o.customerID from customer c
inner join orders o on c.customerID=o.customerID where
c.customername="ghi";

select customername,count(productID) as "total product purchased" from
customer c join orders o join order_product op on
c.customerID=o.customerID and o.orderID=op.orderID and
c.customername="def";

select c.customerID,customername,orderID,o.customerID from customer c
left join orders o on c.customerID=o.customerID and c.customername="ghi";

select c.customerID,customername,orderID,o.customerID from customer c
right join orders o on c.customerID=o.customerID and
c.customername="ghi";

select c.customerID,customername,orderID,o.customerID from customer c
inner join orders o on c.customerID=o.customerID and
c.customername="ghi";

select c.customerID,customername,orderID,o.customerID from customer c
join orders o on c.customerID=o.customerID and c.customername="ghi";

select c.customerID,customername,orderID,o.customerID from customer c
left join orders o on c.customerID=o.customerID;

select customername,count(productID) as "total product purchased" from
customer c join orders o join order_product op on
c.customerID=o.customerID and o.orderID=op.orderID;

select customername,count(productID) as "total product purchased" from
customer c inner join orders o inner join order_product op on
c.customerID=o.customerID and o.orderID=op.orderID;
```

```
select * from customer c inner join orders o inner join order_product op on
c.customerID=o.customerID and o.orderID=op.orderID;

select customerID from orders group by customerID having
count(orderID)>2;

select customerID from orders group by customerID having
count(orderID)>1;

select customerID from orders group by customerID having
count(orderID)>=2;



select count(orderID),orderdate from orders o, customer c where
o.customerID=c.customerID and c.customername="ghi";

select c.customerID,customername,customeraddress,customermobno from
customer c,orders o where c.customerID=o.customerID;

select c.customerID,customername,customeraddress,customermobno from
customer c,orders o where c.customerID=o.customerID and orderID="o1";

select sum(purchase_quantity*price) from order_product op,products p where
op.productID=p.productID and orderID="o1";

select orderID,sum(purchase_quantity*price) from order_product op,products
p where op.productID=p.productID group by orderID;

select orderID,sum(purchase_quantity*price) as totalp from order_product
op,products p where op.productID=p.productID group by orderID order by
totalp;



//samples
```

```
select count(orderID) from orders o, customer c where
o.customerID=c.customerID;

select c.customerID,count(orderID) from orders o, customer c where
o.customerID=c.customerID;

select c.customerID,count(orderID) from orders o, customer c where
o.customerID=c.customerID group by c.customerID;

select c.customerID,count(orderID) as total_orders from orders o, customer c
where o.customerID=c.customerID group by c.customerID order by
total_orders;

select c.customerID,count(orderID) as total_orders from orders o, customer c
where o.customerID=c.customerID group by c.customerID order by
total_orders desc;

select c.customerID,c.customername,count(orderID) as total_orders from
orders o, customer c where o.customerID=c.customerID group by
c.customerID order by total_orders desc limit 1;
```

PL/SQL Block

```
DELIMITER //

CREATE PROCEDURE GetAllProducts()
BEGIN
        SELECT *  FROM products;
END //

DELIMITER ;


//Syntax
CREATE PROCEDURE procedure_name(parameter_list)
BEGIN
```

```
  statements;
END //

CALL stored_procedure_name(argument_list);

Call GetAllProducts;
Call GetAllProducts();

DROP PROCEDURE [IF EXISTS] stored_procedure_name;

DROP PROCEDURE IF EXISTS stored_procedure_name;

DROP PROCEDURE stored_procedure_name;


//Parameter

DELIMITER //

CREATE PROCEDURE GetCustomerByCity(
        IN cityName VARCHAR(255))
BEGIN
        SELECT *
        FROM customer
        WHERE customeraddress= cityName;
END //

DELIMITER ;

call GetCustomerByCity("Kolhapur");


DELIMITER //

CREATE PROCEDURE GetCustomerByCity(
        IN cityName )
BEGIN
```

```
        SELECT *
        FROM customer
        WHERE customeraddress= cityName;
END //

DELIMITER ;



Delimiter $$

CREATE PROCEDURE GetOrderCountByStatus (
    IN  orderStatus VARCHAR(25),
    OUT total INT)
  BEGIN
   SELECT COUNT(orderID)
   INTO total
   FROM orders
   WHERE customerID = orderStatus;
  END$$

DELIMITER ;

call GetOrderCountByStatus(3,@total1);

select @total1;

select @total1 as “total order placed”;


DELIMITER $$

CREATE PROCEDURE SetCounter(
        INOUT counter INT,
    IN inc INT
)
BEGIN
```

```
    SET counter = counter + inc;
END$$

DELIMITER ;

SET @counter = 1;
CALL SetCounter(@counter,1);
CALL SetCounter(@counter,1);
CALL SetCounter(@counter,5);
SELECT @counter;


CREATE TABLE IF NOT EXISTS customer_backup (
customerID INT UNSIGNED NOT NULL AUTO_INCREMENT,
customername CHAR(30) NOT NULL DEFAULT '',
customeraddress VARCHAR(30) NOT NULL DEFAULT '',
customermobno INT UNSIGNED NOT NULL,
PRIMARY KEY (customerID)
);

CREATE TRIGGER trigger_name
{BEFORE | AFTER} {INSERT | UPDATE| DELETE }
ON table_name FOR EACH ROW
trigger_body;

create trigger customer_delete before delete on customer for each row
insert into customer_backup values
(old.customerID,old.customername,old.customeraddress,old.
customermobno);

create trigger product_quantity after insert on order_product for each
row update product set quantity=quantity-new.purchase_quantity where
productID=new.productID;

show triggers;
```