- Embedded Queries

    1. Query to display all warehouses and batches stored in them along with their information

    ```
    Select w.warehouse_id,w.pincode,b.batch_id,b.product_id,
    b.quantity from warehouse w,warehouse_batch wb,batch b
    where w.warehouse_id=wb.warehouse_id and
    b.batch_id=wb.batch_id;
    ```

    2. To show the bill and its details for a particular store

    ```
    select
    bill.*,bill_product.product_id,bill_product.quantity from
    bill left join bill_product on
    bill.bill_id=bill_product.bill_id where
    store_id={store_id} order by bill_id".
    ```

- Olap Queries

    1. To get the capacity of the warehouses with hierarchy of address. Ie the caparity in a district, in a city and in a street(Roll_up)

    ```
    select pincode,city,street,sum(capacity) from warehouse
    group by pincode,city,street with rollup having
    grouping(pincode,city,street)<>0 ;
    ```

    2. To get the Profit of a store with respect to their area, ie the porift of a district, city, and street(Roll_up)

    ```
    select
    pincode,city,street,sum(profit),grouping(pincode),grouping(cit
    y),grouping(street) from retailer
    group by pincode,city,street with rollup order by
    grouping(street),grouping(city),grouping(pincode);
    ```

    3. To find the profit of each product between given fixed date(Slicing)

    ```
    select product.product_id,
    (t2.s1*product.mrp-t2.c*product.cost) as yearly_profit
    from product inner join (select sum(batch.quantity) as
    c,batch.product_id as pid,sale.s1 from batch inner
    join(select sum(bill_product.quantity) as s1,product_id
    ```

```
from bill_product group by bill_product.product_id) as
sale on batch.product_id=sale.product_id where
batch.production_date between '2021-01-01' and
'2022-011-01' group by batch.product_id) t2 on
product.product_id=t2.pid;
```

4. To find the report of which store sold what amount of each product. (Pivot)

```
SET @sql=NULL;
set session group_concat_max_len=10000;
SELECT group_concat(Distinct concat(
    'SUM(case when bills.product_id="',bills.product_id,'"
then bills.quantity else 0 end) as ',bills.product_id)
)
INTO @sql
FROM (select store_id,product_id,quantity from
bill_product natural join bill) as bills;

SET @sql=concat('SELECT bills.store_id, ', @sql,' from
(select
bill.store_id,bill_product.product_id,bill_product.quantit
y from bill_product
natural join bill) as bills group by bills.store_id');

prepare stmt from @sql;
execute stmt;
deallocate prepare stmt;
```

5. To find non empty warehouses and the number of batches stored in them. (Having)

```
SELECT warehouse_id, COUNT(batch_id) as num_batches
FROM warehouse_batch GROUP BY warehouse_id HAVING
COUNT(batch_id) > 0;
```

6. To find capacity of warehouses depending on their address(Cube)

```
select pincode,city,street,sum(capacity) from warehouse
group by cube(pincode,city,street);
```

- Triggers

1. A trigger the manage the products that are being added to a bill. This trigger checks if a the particular store has enough quantity of a particular product. If yes, then quantity is removed from the oldest batch. This also updates the total amount in the bill. Else the product is not added to the particular bill.

```
delimiter //
CREATE TRIGGER buying_product BEFORE INSERT ON bill_product
FOR EACH ROW
BEGIN
    IF NOT EXISTS (select * from store_inventory si,bill b
where si.quantity>=new.quantity and
si.product_id=new.product_id and si.store_id=b.store_id and
b.bill_id=new.bill_id) THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Insufficient products';
    ELSE
        update batch inner join store_inventory si on
batch.batch_id=si.batch_id
        inner join bill b on b.store_id=si.store_id and
b.bill_id=new.bill_id
        set batch.quantity=batch.quantity-new.quantity where
        si.quantity>=new.quantity and
si.product_id=new.product_id order by si.expiry_date limit 1;

        update bill set
total_amount=total_amount+new.quantity*(select mrp from product
where product.product_id=new.product_id)
        where bill_id=new.bill_id;

  END IF;
END;
//

Delimiter  ;
```

2. A trigger that manages the batches being stored in warehouses. It checks if the the warehouse has enough empty capacity to store the batch. If yes, the batch will be added to the warehouse, else an error will be raised regarding insufficient space.

```
delimiter //
```

```
CREATE TRIGGER loading_batches
BEFORE INSERT ON warehouse_batch
FOR EACH ROW
BEGIN
    -- Calculate the total quantity of batches in the warehouse
    -- DECLARE total_quantity INT ;
    SELECT SUM(batch.quantity) INTO @total_quantity
    FROM batch
    INNER JOIN warehouse_batch ON batch.batch_id =
warehouse_batch.batch_id
    WHERE warehouse_batch.warehouse_id = NEW.warehouse_id;

    -- Check if the warehouse has enough capacity to store the
batches
    -- DECLARE warehouse_capacity INT;
    SELECT capacity INTO @warehouse_capacity
    FROM warehouse
    WHERE warehouse_id = NEW.warehouse_id;

    -- DECLARE cur_batch INT;
    SELECT batch.quantity into @curbatch from batch where
batch.batch_id=new.batch_id;
    if @total_quantity is NULL Then set @total_quantity=0; end if;
    if @curbatch is NULL Then set @curbatch=0; end if;
    if @warehouse_capacity is NULL Then set @warehouse_capacity=0;
end if;

    IF @total_quantity + @curbatch > @warehouse_capacity THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Warehouse has reached maximum
capacity';
    END IF;
END;

//

delimiter   ;
```