—---VIEWS—-------—--

```
-- order_id | order_date | customer_id | first_name | points |
-- product_id | product_name | unit_price | quantity

use sql_store;

select * from orders;
select * from customers;
select * from order_items;
select * from products;

select o.order_id, o.order_date, o.customer_id, c.first_name, c.last_name,
        c.points, p.product_id, p.name, p.unit_price, p.quantity_in_stock
from orders o
join customers c
on o.customer_id = c.customer_id
join order_items oi
on o.order_id = oi.order_id
join products p
on oi.product_id = p.product_id;


create view orders_customers_orderitems_products
as select o.order_id, o.order_date, o.customer_id, c.first_name, c.last_name,
        c.points, p.product_id, p.name, p.unit_price, p.quantity_in_stock
from orders o
join customers c
on o.customer_id = c.customer_id
join order_items oi
on o.order_id = oi.order_id
join products p
on oi.product_id = p.product_id;

select * from orders_customers_orderitems_products;

-- Generally views should be read    only.

create view o_c_oi_p
as select o.order_id as id, o.customer_id as c_id, c.first_name as name, p.name as product
from orders o
join customers c
on o.customer_id = c.customer_id
join order_items oi
```

```
on o.order_id = oi.order_id
join products p
on oi.product_id = p.product_id;

select * from o_c_oi_p;

start transaction;

update orders_customers_orderitems_products
set first_name = 'Shyam';

rollback;

-- Views are updatable, but it will not only udpate the view but it will also update the underlying
table.

start transaction;

update orders_customers_orderitems_products
set first_name = 'Shyam', name = 'chocolate';

rollback;

-- Views doesn't allow to udpate more than one base table in 1 query.

-- Error Code: 1393. Can not modify more than one base table through a join view
'sql_store.orders_customers_orderitems_products'

start transaction;

update orders_customers_orderitems_products
set phone = 123;

rollback;

-- Error Code: 1054. Unknown column 'phone' in 'field list'



—-------------------------------------WINDOW FUNCTIONS—-------------------------------------------------
SELECT * FROM sql_store.student;

-- Get the id of all the students along with the avg iq.
```

```
select id, avg(iq)
from student;

-- Error Code: 1140. In aggregated query without GROUP BY, expression #1 of SELECT list
contains nonaggregated column 'sql_store.student.id'; this is incompatible with
sql_mode=only_full_group_by

-- id | avg iq

-- subquery
select id, (select avg(iq) from student)
from student;

--  this will not give us the right data.
Select id, avg(iq) from Student
group by id;

-- Window Function
select  id,
                    avg(iq) OVER()
from student;

-- Using OVER(), we'll be able to retain the information regarding the original table along with
the aggregate function that is not possible without window function.

-- Get the id of all the student along with the avg iq of all the students of their batch.

-- 1.
select id, (select avg(iq) from student s2 where s2.batch_id = s1.batch_id)
from student s1;

-- 1 -> avg iq of all the students with batch_id = batch_id of id=1
-- stud1 -> avg iq of all the students who belongs to same batch as stud1

-- 2.
select id, (select avg(iq) from student s2 group by batch_id having s2.batch_id = s1.batch_id)
from student s1;

-- 3. Window Function
-- GROUP BY --> PARTITION BY in window function.
select  id,
                    avg(iq) OVER(PARTITION BY batch_id)
from student;
```

```
-- Get the name of all the students with their rank by iq.

select id, name
from student
order by iq desc;

-- RANK FUNCTION;
select id, name, RANK() OVER(ORDER BY iq DESC)
from student;

-- RANK FUNCTION;
select id, name, RANK() OVER(ORDER BY iq DESC, name ASC)
from student;

-- 1,1,3,3,3,3, 7, ...  => SPARSE RANK

 -- DENSE_RANK
select id, name, dense_rank() OVER(ORDER BY iq DESC) as iq_rank
from student;
-- where iq_rank = 2;

-- ROW_NUMBER
select id, name, ROW_NUMBER() OVER(ORDER BY iq DESC) as student_rank
from student;



-- 1, 1, 2, 2, 2, 3, 3, 3, 3 4, .....
```