In [ ]:

```
-- is used for comment in mysql # after double desh there should be space
```

In [2]:

```
USE sql_store # to use database
```

In [ ]:

```
SELECT *
from customers; # selectiing all from customer table
```

In [ ]:

```
SELECT *
from customers
where customer_id = 1
order by first_name
```

In [ ]:

```
# order is must while writing query

select
from
where
order by
```

The SELECT CLAUSE

In [ ]:

```
Select *
from customers # will retrun everything from customers table
```

In [ ]:

```
select last_name,first_name,points
from customers  # will select only three columns from customer tabels
```

In [ ]:

```
select last_name,first_name,points, (points*10) as new_point
from customers  # making a new column from points using alias new_point
```

In [ ]:

```
# if line is too long then we can write above command as below

select
    last_name,
    first_name,
    points,
    (points*10)+100 as new_point
from customers

# always remember in math (/ and *) high priority so use prantheses for desired result
```

DISTINCT KEYWORD

In [ ]:

```
select distinct state
from customers #  will return all disticnt keyword from state column
```

In [ ]:

```
#
select
    name,
    unit_price,
    unit_price * 1.1 as new_price #(increasing all products by 10 percent)

from products

```

WHERE CLAUSE

In [ ]:

```
# we use where clause to filter the dat
select *

from customers
where points >3000 # only data where points coulmn value is greater than 3000
```

In [ ]:

```
select *

from customers
where state = "va" # filter the data where state is va (or also we can use <> operator
```

In [ ]:

```
select *

from customers
where birth_date> "1990-01-01"
```

AND OR and NOT operators

In [ ]:

```
select *
from customers
where birth_date> "1990-01-01" and points >1000 # both condition should be true
```

In [ ]:

```
select *
from customers
where birth_date> "1990-01-01" or
      (points >1000 and state = "va")
```

In [ ]:

```
select *
from customers
where not (birth_date >"1990-01-01" or points >1000)
```

In [ ]:

```
select *
from order_items
where order_id = 6 and (quantity*unit_price) >30

# select from order_items table where order_id is 6 and total bill is greater than 30
# note total bill =(quantity*unit_price)
```

In [ ]:

```
select *
from customers
where state = "va" or state = "ga" or state = "fl"
```

In [ ]:

```
# as we can see in above query we are selection states from va or ga or fl
# we can use in operator
# now we write above query in other format as well lets see how

select *
from customers
where state in("va","fl","ga")
```

In [ ]:

```
# if we want to avoid above three states and want all other states
select *
from customers
where state not in("va","fl","ga")
```

In [ ]:

```
select *
from products
where quantity_in_stock in(49,38,72)
```

THE BETWEEN OPERATOR

In [ ]:

```
select *
from customers
where points >=1000 and points<=3000

# above query we can write with the help of Between operator
# lets try this
```

In [ ]:

```
select *
from customers
where points between 1000 and 3000 #(it will also inculse 100 and 3000)
```

In [ ]:

```
# finding the people between  "1990-01-01" and "2000-01-01"
select *
from customers
where birth_date between "1990-01-01" and "2000-01-01"
```

LIKE OPERATOR

In [ ]:

```
#all customers whose last name starts wiht b
select *
from customers
where last_name like "b%"
```

In [ ]:

```
# we can write any number of charcter
select *
from customers
where last_name like "brush%"
```

In [ ]:

```
# we can aslo find the customers contain b anywhere in there last name
select *
from customers
where last_name like "%b%"
```

In [ ]:

```
# customers whose last name ends with y

select *
from customers
where last_name like "%y"
```

In [ ]:

```
# customers whose last name ends with y. also the have only five charcter before y
select *
from customers
where last_name like "_____y" # because of 5 underscores
```

In [ ]:

```
# start with b and ends with y. also y character between b and y
select *
from customers
where last_name like "b____y"  # because of 4 underscores
```

In [ ]:

```
# % for any number of characters
# _ singel character
```

In [ ]:

```
#Question: Get the customers address whose
#1. address contain Trail or Avenue
#2. Phone numbers not end with 9
```

In [ ]:

```
#1. address contain Trail or Avenue
select *
from customers
where address like "%TRAIL%" Or address like "%AVENUE%"
```

In [ ]:

```
#2. Phone numbers not end with 9
select *
from customers
where phone not like "%9"
```

REGEXP

In [ ]:

```
select *
from customers
where last_name like "%field%"
```

In [ ]:

```
# we use regular expression for more complex query. Above example can be written with t
#lets see how

```

In [ ]:

```
select *
from customers
where last_name regexp "field" # for above query this time we did not use %%
```

In [ ]:

```
# last name ends with field
select *
from customers
where last_name regexp "field$"
```

In [ ]:

```
# last name starts with field
select *
from customers
where last_name regexp "^field" # carrot sign

```

In [ ]:

```
# contain field or mac

select *
from customers
where last_name regexp "field|mac"
```

In [ ]:

```
#contain field or mac or rose
select *
from customers
where last_name regexp "field|mac|rose"
```

In [ ]:

```
# can start with field or contain mac or rose
select *
from customers
where last_name regexp "^field|mac|rose"
```

In [ ]:

```
# can ends with field or contain mac or rose
select *
from customers
where last_name regexp "field$|mac|rose"
```

In [ ]:

```
# last name should contain ge or ie or me anywhere
select *
from customers
where last_name regexp "[gim]e"
```

In [ ]:

```
# range of characters
select *
from customers
where last_name regexp "[a-h]e"
```

In [ ]:

```
#even we can specify characters
select *
from customers
where last_name regexp "[abcdefs]e"
```

In [ ]:

```
#^ biginnning
# $ end
# |logical or
#[a-t] range
#[abcd] spicify character
#
```

In [ ]:

```
# exersize for you
# Get the customers------>
#1. whose first names are elka or ambur
#2. whose last names end with ey or on
#3.last names start with my or contains se
#4. last names contain b followed by r or u
```

In [ ]:

```
#1. whose first names are elka or ambur
select *
from customers
where first_name regexp "elka|ambur"
```

In [ ]:

```
#2. whose last names end with ey or on
select *
from customers
where last_name regexp "ey$|on$"
```

In [ ]:

```
#3.last names start with my or contains se
select *
from customers
where last_name regexp "^my|se"
```

In [ ]:

```
#4. last names contain b followed by r or u
select *
from customers
where last_name regexp "[ru]b"
```

IS NULL OPERATOR

In [ ]:

```
# find all the data where customer phone number is null

select *
from customers
where phone is null
```

In [ ]:

```
# find all the data where customer phone number is not null
select *
from customers
where phone is not null
```

In [ ]:

```
#exersize get the orders that are not shipped

use sql_store; # as this data is available in sql_store database
select*
from orders
where shipped_date is null and shipper_id is null
```

ORDER BY CLAUSE

In [ ]:

```
# sorting the table with the first name of customers
select*
from customers
order by first_name
```

In [ ]:

```
# also we con do it in descinding order
select*
from customers
order by first_name desc
```

In [ ]:

```
select*
from customers
order by state,first_name # if two customers are from same state than it will sort thos
```

In [ ]:

```
select*
from customers
order by state desc,first_name desc
```

In [ ]:

```
select first_name,last_name,birth_date
from customers
order by birth_date
```

In [ ]:

```
# above code can be written as below
select first_name,last_name,birth_date
from customers
order by 1,2  # 1 refers to first_name and 2 refers to last_name but not recommended
```

In [ ]:

```
select *,quantity*unit_price as total_bill
from order_items
where order_id = 2
order by quantity*unit_price desc
```

In [ ]:

```
# as we can see above we needed to use quantity*unit_price two times
# lets do this by using total_bill only
```

In [ ]:

```
select *,quantity*unit_price as total_bill
from order_items
where order_id = 2
order by total_bill desc
```

LIMIT CLAUSE

In [ ]:

```
select *,quantity*unit_price as total_bill
from order_items
where order_id = 2
order by total_bill desc
limit 2   # will show only two results
```

In [ ]:

```
select *
from customers
limit 6,3 # skip first 6 records and pick pick 3 records
```

In [ ]:

```
# Exersize get three loyal customers (customers who has maximum points)

select *
from customers
order by points desc
limit 3  # limit cluase always come in the end alway pay attention to order
```

INNER JOINS

In [ ]:

```
# in inner joins (inner is optional)
select *
FROM orders
join customers on orders.customer_id = customers.customer_id
```

In [ ]:

```
select first_name,last_name
FROM orders
join customers on orders.customer_id = customers.customer_id
```

In [ ]:

```
# if we want to display customer id (it will throw an error that )
#error customer_id in this field is ambiguous
# becuse we have customer id in both table
# to avoid this error we need to write this with table name
# we can choose either tabel name because customer id is same in both table
# lets do it in next cell
select customer_id,first_name,last_name
FROM orders
join customers on orders.customer_id = customers.customer_id
```

In [ ]:

```
# so we chose here orders.customer_id
select orders.customer_id,first_name,last_name
FROM orders
join customers on orders.customer_id = customers.customer_id
```

JOINING ACROSS DATABASES

In [ ]:

```
select *
from order_items oi
join sql_inventory.products p
    on oi.product_id = p.product_id
```

In [ ]:

```
# always remember :sql_inventory.products we use sql.invertory database here as we do r
# in other database
```

SELF JOIN

In [ ]:

```
# using the same table to join (in our case we are using employee table only)
use sql_hr;

select *
from employees e
join employees m
    on e.reports_to = m.employee_id

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
```

In [ ]:

```
```