select \* from departments;

select \* from employees;

select \* from regions;

-- select statment in retriving statement

/\* to filter data we use where clause , if you filter character we use ='intended character'

or use like

'can put letter that charcter is starting with%exact part of satement we are looking for %'

% means can be started or followed by anything

if we want to filter numeric data we use logic sings like = < > =< >=\*/

select first\_name , last\_name, salary, department from employees where department like 'C%t%g';

/\* more about filtering and going in more depth, using and and or and cmbining them ,

and is compulsory and or is conditional , each row with be go through the condition and then give the result

for each row.\*/

select \*from employees where last\_name like '%mo%'and department='Clothing';

--pay attention to lower or uuper case

select \*from employees where last\_name like '%mo%'and department='Music';

select \*from employees where salary <> 10000 /\*we ca use where not or != \*/ and

( department='Music' or Gender='F');

select \*from employees where salary <> 10000 /\*we ca use where not or != \*/ and

( department !='Music' or Gender='F');

select \*from employees where salary <> 10000 /\*we ca use where not or != \*/ and

department='Music' or Gender='F';-- look at the line 140 but it is better to seperate with parantises

select \* from employees where not department <> 'Sports';

--Null can not be compared to any thing even itself BY EQUALITY OPERATORS OR ETC.

SELECT \* FROM employees WHERE NULL!=NULL;

-- HOW TO WOTK WITH NULL>>> USING 'IS'

SELECT \*

FROM employees

WHERE email IS NULL;-- IS NOT OR WHERE NOT IS

SELECT \*

FROM employees

WHERE email IS NOT NULL;

SELECT \*

FROM employees

WHERE NOT email IS NULL;

--WHAT IF WE HAVE MULTINPE SAME CONDITIONS LIKE

--SELECTING RANGE FOR CHARACTERS

SELECT \*

FROM employees

WHERE department='Sports',

department='Toys',

departmet='Garden',

-- do we need to

--write all three times department and =?

--here we can use 'IN' STATEMENT.

SELECT \*

FROM employees

WHERE department IN ('Sports',

'Toys',

'Garden');

-- SELECTING RANGE FOR INTEGERS

SELECT \*

FROM employees

WHERE salary BETWEEN 80000 AND 100000;

--Exercise one:

SELECT first\_name, email

FROM public.employees

WHERE GENDER='F'

AND department='Tools'

AND salary>110000;

SELECT first\_name, hire\_date

FROM public.employees

WHERE salary>165000

Or (GENDER='M'

AND department='Sports');

SELECT first\_name,hire\_date

FROM public.employees

WHERE hire\_date

Between '2002-01-01'

and '2004-01-01';

SELECT first\_name,hire\_date

FROM public.employees

WHERE hire\_date < '2002-01-01'

OR hire\_date >'2004-01-01';

SELECT \*

FROM public.employees

WHERE gender='M'

AND department='Automotive'

AND salary > 40000

AND salary < 100000

Or(gender='F' and department= 'Toys' );

---order by , limit distinct and remaing column

-- sorting>>> 'order' clause

-- where is following with condition

-- but order by is not realted to filtering

--and is not realted to where

-- order by is 'asc' by default

-- if you need vis versa you need

--to add 'desc' even for alphabaticly

Select \* from public.employees

ORder by Department desc;

Select \* from public.employees

ORder by Salary ;

--having unique value in column we can use

-- 'distinct' clause

Select distinct department

from public.employees;

-- we can sorting the lise alphabaticlly

Select distinct department

from public.employees

order by Department ;-- or we can use 1

Select distinct department

from public.employees

order by 1;

-- having op 10 in this department : limit #

Select distinct department

from public.employees

order by 1

Limit 10;

-- we can use 'fetch' instead of limit

-- 'Fetch first # rows only'

Select distinct department

from public.employees

order by 1

Fetch first 3 rows only;

-- we can name our retive

--data column anything we want

-- using ' as " New Name" '

Select first\_name as "surname"

from public.employees;

--formating data,

--combine data, uppercase,

--find lenght , or

--take part of data and manupulate it

SELECT UPPER(first\_name), LOWER(last\_name),

LENGTH/\*usulful form cleaningdata\*/ (department)

FROM public.employees;-- there are for report perpose

--forexample deleting extra space in data

--how to test functions

SELECT ' HELO THERE '

SELECT TRIM(' HEllo there')

SELECT length (TRIM(' HEllo there'))

--creat new column , combining two column

--using ||

SELECT first\_name || ' ' || last\_name as combine

FROM public.employees;

-- another way to creat new column with

--existing column is using boolean expression

SELECT first\_name || ' ' || last\_name

as full\_name, (salary >140000)

FROM public.employees

order by salary desc;

Select department,

('Clothing' in (department))--boolean statement is always is column of T/F

FROM public.employees;

Select department,

(department like '%oth%')--boolean statement is always is column of T/F

FROM public.employees;

Select count (department)FROM public.employees;

-- function to extract data from existing data

--using substring, replace , position, coalesce

SELECT 'this is a test data' "test data";--if there is a space

-- we put in "" otherwise no need any thing

-- there are two way to write substring function

--one there is a range :

SELECT SUBSTRING('this is a test data'

FROM 1 for 4) test\_data\_extracted;

-- there is just start point :

SELECT SUBSTRING('this is a test data' ,

4) test\_data\_extracted;

SELECT department,

Replace (Department, 'Clothing', 'dress') newname

from public.employees;

SELECT department,

(Department || ' Department') newdepartment

from public.employees;

SELECT email, substring (email, position('@' in email)+1)

FROM employees;

-- cleaning data which ar null and putting a value or sth in the cell

Select COALESCE (email, 'None') new\_email\_field

from public.employees;

--the above functions using

--reformating data especialy character, runing one every single cell

--grouping function

--1- Max- MIN-AVG-Round- COUNT-SUM

SELECT MAX(Salary) --every single row send to MAx and when through the function.

--but we have one output.

FROM public.employees;

SELECT AVG(Salary)

FROM public.employees;

SELECT Count(employee\_id)--counting from 1 for each existing data in a column

FROM public.employees;

SELECT Count(1)--counting for each recorde

FROM public.employees;

SELECT Count(\*)--counting for each recorde

FROM public.employees;

SELECT Count(email)

FROM public.employees;

--to count how many of same data do we have in the column

--we use count function with group by function

Select count(1), department

from public.employees

group by department;

Select count(1)--same result as above

--frist going to employees table

--and grouping then counting for each group

from public.employees

group by department;

-- Questin, count how many employee using same domain for their email

Select count(1) as numebr\_of\_each\_domain, substring(email, position('@' in email)) as same\_domain from public.employees

where not email is Null

group by same\_domain;

--having max(numebr\_of\_each\_domain); as we can see we can not do this

--so

select max(numebr\_of\_each\_domain)

from (Select count(1) numebr\_of\_each\_domain,

substring(email, position('@' in email)) as same\_domain

from public.employees

where not email is Null

group by same\_domain) as newtable;

-- so now we know there is an email omin which

--is used 7 times

--??but which one?

Select count(1) numebr\_of\_each\_domain,

substring(email, position('@' in email)) as same\_domain

from public.employees

where not email is Null

group by same\_domain

having count(\*)>6

-- order by count(\*) desc;

-- how to use MAX function for Count function in last qiestion.

--?most popular domain

select \* from public.employees;-- you can see here Same\_domin is not created in the main table

--where clause is limit the recors and group by is coming after where

select department , sum(salary) from public.employees

Where region\_id in (4,5,6,7)

Group by department ;

select department, region\_id , sum(salary) from public.employees

Where region\_id in (4,5,6,7)

Group by department, region\_id

order by department ;

-- we can not have feature or column

--when we have group by and tha cloumn

--is not in aggregation function or in group by clause

--

-- we could have no where stament

Select department , count(1) total\_number\_of\_employees,

round(AVG(salary))

from employees

group by department

order by total\_number\_of\_employees desc;

-- very importnat

-- where clause is used to

--filter recoreds not filter aggredated data

-- in this case we should use 'having ' caluse

--which comes after group by and before order by

Select department, count(\*)

from employees

group by department

having count(\*)>35

order by department;

Select first\_name, count(\*)

FROM public.employees

group by first\_name

having count(\*)>2-- in having clause we can just have boolean argument

;

--q--show the unique departement without using distinct statement

select distinct department from employees;

select department from employees group by department;

select gender, region\_id,

Min(salary) Min\_salary,

MAx(salary) Max\_salary, round(AVg(salary)) Avg\_salary

from employees

group by gender, region\_id

order by gender , region\_id,Min\_salary,

Max\_salary, Avg\_salary asc;

--referencing in columns name by using table name or giving

--alianses to the tables

--by putting name infront and using that alianses

--table are the source of data.

--subquaries

--select statement is another select statement

select \* from employees

where department not in

(select department from departments)-- this select part is series of data

-- using the subqueries in the where clause

-- we can use in the from clause.

select \*

from ( select \* from employees where salary> 150000) a;

-- a is alias es for the table

--if you put unqueries we need alias

select a.first\_name

from ( select \* from employees where salary> 150000) a;

select \* from employees

where department in

(select department from departments)

--subquaries can be in the select clause but

--\*\*\*\* we should notice the result of select satatement is

-- series of data and rows so the concept f column is comming after each row

--therefore, the below statment is not legal

select first\_name,last\_name, salary (select first\_name from employees)

from employees;

--this query will go to the employees table satring from

--first row (highlighting first row) then selecting aforementioned cloumn

--selecting the cell of first\_name, last\_name, salary but iin the

--next cell we have lots of rows then the sql faces a problen to selct which row of last cell

-- \*\*\* so we can limit the second select statement to a single data

select first\_name,last\_name, salary,

(select first\_name from employees limit 1)

from employees;

select first\_name from public.employees

where department in

(select department from public.departments

where division='Electronics' );

select \* from public.employees

where salary >30000 and region\_id in (4,5,6,7);

select \* from public.employees

where salary >30000 and region\_id in ( select region\_id from public.regions where country= 'Asia' or country= 'Canada')

-- we can write: where country in ('Asia' , 'Canada')

\_\_\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FANTASTIC QUESTION MAKE BY ME AND SLOVED BY ME\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

select first\_name, department , (select Maxsalary

from (select department, Max(salary) Maxsalary

from employees group by department ) b

where a.department= b.department)-salary

from employees a

WHere region\_id in ( select region\_id from public.regions where country= 'Asia' or country= 'Canada')

;

-- sql going to the select and jumping to the from part recognize the table

-- then highlight the recordes with specific column and if a cell is not ready calculating that

--even if there is an another select clause \*\*the first step of each clause is rcognizing table and then conditions

--cell afetr being ready

-- jumping to where clause and checking the conditios.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--comparing one Value by multiplate value using ANY and All

select \* from employees

where region\_id in

(select region\_id from regions

where country='United States');

-- is equal to this quary as will

--compare each cell with the one we have like loop

select \* from employees

where region\_id= any--is like or and All act as and

(select region\_id from regions

where country='United States');

select \* from employees

where region\_id>any--is like or and All act as and

(select region\_id from regions

where country='United States');

--look this quary

select \* from employees

where region\_id= all--is like or and All act as and

(select region\_id from regions

where country='United States');

-- we can use any and all with having clause

--Quary:

--write a quary that returns all of

--those employees that work in the kids

--division and the dates at which

--those employees were hired is greater than all of

--the hire\_date of employees who work in

--the maintenamce department

select \* from employees

where department= any (

select department

from Departments

where division='Kids')

and (hire\_date > all

(select hire\_date

from employees

where department='Maintenance'))

--quary

--those salary that appears most frequently

select salary, count(\*)

from employees

group by salary

order by count(\*) desc

limit 1;

--or

select salary, count(\*)

from employees

where salary>= any (select salary from employees

group by salary

order by count(\*) desc

limit 1;

Create Table dupes (id integer, name varchar(10));

insert into dupes Values (1, 'frank');

insert into dupes Values (2, 'frank');

insert into dupes Values (3, 'robert');

insert into dupes Values (4, 'robert');

insert into dupes Values (5, 'sam');

insert into dupes Values (6, 'frank');

insert into dupes Values (7, 'petter');

--Select unique Value,

select distinct name, id

from dupes;--this will consider a pair of data and that

--is why introduce each one as a unique

select name ,Min(id)

from dupes

group by name;

--if we want the column

select \* from dupes where id in (

select Min(id)

from dupes

group by name);

--delete dups fro table

delete from dupes where id not in (

select Min(id)

from dupes

group by name);

select \*from dupes;

--learning delete and update on your own

--delete table premanently;

drop table dupes;

Select\* from Employees;

--find out outlayer

Select round(Avg(salary)) from employees

where Salary not in(

(select Max(Salary) from employees),

(select Min(Salary) from employees))

-- Conditional Expession

--create new column to display

--who over paid or under paid

Select first\_name, salary,

case

when salary>100000 then 'paid well'

when salary<100000 then 'under paid'

else 'unpaid'

end

from employees

order by salary desc;

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Select first\_name, salary,

case

when salary<100000 then 'under paid'

when salary>100000 and salary<160000 then 'paid well'

when salary>160000 then 'executive'

else 'unpiad'

end as category

from employees

order by salary desc;

--\*\*\*\*\*\*\*\*\*\*\*\*

select a.category, count(\*) from(

Select

case

when salary<100000 then 'under paid'

when salary>100000 and salary<160000 then 'paid well'

when salary>160000 then 'executive'

else 'unpiad'

end as category

from employees)a -- when you are using source of data that are not table you needto give aliase

group by a.category;

--Transpose the data turning rows to columns

--using sum and case function

select sum(underpaid) underpaid , sum(paidwell) paidwell, sum(executive) executive from(

select

(case

when salary <100000 then 1

end) as underpaid,-- case act as column here you need to sperate them with,

(case

when salary>100000 and salary<160000 then 1

end) as paidwell,

(case

when salary>160000 then 1

end) as executive

from employees

group by salary) a

;---- you can have another way

select

sum (case

when salary <100000 then 1

end) as underpaid,-- case act as column here you need to sperate them with,

sum (case

when salary>100000 and salary<160000 then 1

end) as paidwell,

sum(case

when salary>160000 then 1

end) as executive

from employees;-- we do not need to put else here as a defult it will consider null for others but with putting else 0

--we can put an integer value in the cell

select

sum (case

when salary <100000 then 1 else 0

end) as underpaid,-- case act as column here you need to sperate them with,

sum (case

when salary>100000 and salary<160000 then 1 else 0

end) as paidwell,

sum(case

when salary>160000 then 1 else 0

end) as executive

from employees;

---\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--transpose

select count(case when department ='Sports' then 1 end)as Sport,

count(case when department ='Tools' then 1 end)as Tools,

count(case when department ='Clothing' then 1 end)as Clothing,

count(case when department ='Computers' then 1 end )as Computers

from employees;

select first\_name,

(case when region\_id =1 then 'United States' end)as region\_1,

(case when region\_id =2 then 'United States' end)as region\_2,

(case when region\_id =3 then 'United States' end)as region\_3,

(case when region\_id =4 then 'Asia' end)as region\_4 ,

(case when region\_id =5 then 'Asia' end)as region\_5 ,

(case when region\_id =6 then 'Canada' end)as region\_6 ,

(case when region\_id =7 then 'Canada' end)as region\_7

from employees

order by first\_name;

select first\_name,

case when region\_id =1 then 'United States'

when region\_id =2 then 'United States'

when region\_id =3 then 'United States'

when region\_id =4 then 'Asia'

when region\_id =5 then 'Asia'

when region\_id =6 then 'Canada'

when region\_id =7 then 'Canada'

end

from employees

order by first\_name;

---select from region table

select \* from regions;

select first\_name , (case when region\_id=1 then (select country from regions where region\_id=1 ) end ) as region\_1,

(case when region\_id=2 then (select country from regions where region\_id=2 ) end ) as region\_2,

(case when region\_id=3 then (select country from regions where region\_id=3 ) end ) as region\_3,

(case when region\_id=4 then (select country from regions where region\_id=4 ) end ) as region\_4,

(case when region\_id=5 then (select country from regions where region\_id=5 ) end ) as region\_5,

(case when region\_id=6 then (select country from regions where region\_id=6 ) end ) as region\_6,

(case when region\_id=7 then (select country from regions where region\_id=7 ) end ) as region\_7

from employees;

--\*\*\*\*\*\* Joint Tables can make it easier to pull out information from two table

--not using joint

select count ( case when region\_id in (1,2,3) then 1 end) as "United States" ,--"" as we have space

count ( case when region\_id in (4,5) then 1 end) as Asia ,

count ( case when region\_id in (6,7) then 1 end) as Canada

from employees;

---or

select count(a.region\_1)+count(a.region\_2)+count(a.region\_3) as "United States" ,

count(a.region\_4)+count(a.region\_5) as Asia ,

count(a.region\_6)+count(a.region\_7) as Canada

from(

select first\_name , case when region\_id=1 then (select country from regions where region\_id=1 ) end as region\_1,

case when region\_id=2 then (select country from regions where region\_id=2 ) end as region\_2,

case when region\_id=3 then (select country from regions where region\_id=3 ) end as region\_3,

case when region\_id=4 then (select country from regions where region\_id=4 ) end as region\_4,

case when region\_id=5 then (select country from regions where region\_id=5 ) end as region\_5,

case when region\_id=6 then (select country from regions where region\_id=6 ) end as region\_6,

case when region\_id=7 then (select country from regions where region\_id=7 ) end as region\_7

from employees) a;

--------------------------------correlated subquary means sub quary take sth from out

select first\_name, salary --for each record condition and inner quary are run

from employees e1

where salary> (select round (avg(salary))

from employees e2 --\*\*\*\* needed one cell to be selected so we need to use where

where e1.department= e2.department

group by department)

------------------------------------------

select department ,count(employee\_id) from employees

group by department

having count(employee\_id)> 38

---- aggregation function can not be used in where clause---

select department, (select count(employee\_id) from employees a

where a.department =b.department)

from employees b

where (select count(employee\_id) from employees a

where a.department =b.department) > 38

group by department;

/\*run the below QUARY STEP BY STEP FRIRST GOING TO THE MAIN TABLE

SECOND READING THE FIRST ROW AND CELL IF THERE IS CLULATINON IS DOING IT

THIRD CHEAKING CONDITION FOR EACH ROW

FORTH RENAME IF THERE IS A COMMAND

....\*/

select department, (select count(employee\_id) from employees a

where a.department =b.department)as number\_of\_employees

from employees b

--where number\_of\_employees

/\*select department, (select count(employee\_id) from employees a

where a.department =b.department) as number\_of\_employees

from employees b

where number\_of\_employees > 38-- this will not work because before

--calculating the

--number of employee it comes to

--fisrt codition to check for

--the record like department

--so this cell should be calulated in where like above or

-- usedin where"

-- which is not possible as aggrigation function

--can not be usd in where claused

select department, (select count(employee\_id) from employees a

where count(employee\_id)>38 )

from employees b

-- is an aggregation function \*/

select department,(select max(salary) from employees where department =b.department)

from departments b

where (select count(employee\_id) from employees a

where a.department =b.department) > 38

;

--------------------------Exercise

select \* from(

select department, first\_name,

case when salary = (select Max(salary)from employees b where a.department= b.department) then Max(salary)

when salary = (select min(salary)from employees b where a.department= b.department) then min(salary)

end salaryy,

case when salary = (select Max(salary)from employees b where a.department= b.department) then 'highpaid'

when salary = (select min(salary)from employees b where a.department= b.department) then 'lowpaid'

end salary\_in\_department

from employees a

group by department, first\_name, salary)d

where salaryy is not null-- we could using comparison between salary and min and max

group by department,d.first\_name,d.salaryy,d.salary\_in\_department ;--instead we could use order by

;

-------------------------

select department,first\_name,salary ,

case when salary = maxi then 'highpaid'

when salary = mini then 'lowpaid'

end salary\_in\_department

from(

select department, first\_name, salary,

(select Max(salary)from employees b where a.department= b.department) maxi,

(select min(salary)from employees b where a.department= b.department) mini

from employees a

group by department, first\_name, salary)d

where salary= maxi or salary= mini

order by department;

-------------------------------------------------------------------------

-----------JOIN

--pull out data from different sources

select first\_name, country

from employees a, regions b

where a.region\_id= b.region\_id

select first\_name, email, division, country, employees. department

from employees, departments, regions

where employees.department=departments.department and employees.region\_id= regions.region\_id

and email is not null;

-----------SYNTAX for join

select first\_name, country

from employees a inner join regions b

on a.region\_id= b.region\_id;

-----------------

select first\_name, email, division

from employees a inner join departments b

on a.department= b.department ;

----------\*\*\*\*\*\*inner joint just giving the matches of

---both table including left or right

select first\_name, email, division,country

from employees a inner join departments b

on a.department= b.department

inner join regions d on a.region\_id= d.region\_id-- this join the joint of employees and departments

where email is not null;

----------------------------

select distinct department from departments ;

select distinct department from employees;

-------------------------------

select distinct department from employees a

where

a.department not in (select distinct department

from departments);

--------------------------------------------

------\*\*\*left join give the

-------preference to the left

-------right join give the preference to the right table

------they call it outer join

select distinct employees.department employees\_department,

departments.department departments\_department

from employees left join departments

on

employees.department = departments.department;--giving 27 equal to the employees.department

----------- --we should have 3 null

select distinct employees.department employees\_department,

departments.department departments\_department

from employees right join departments

on

employees.department = departments.department;-----giving 24 equal to the employees.department

-- we have one null

---------------------

select distinct employees.department

from employees left join departments

on

employees.department = departments.department

where departments.department is null

--where employees.department not in (select distinct department

from departments)

;

---- right outer join , left outer join , full outer join

--full outer join cosider both left and right giving all matches left and right preferences

----------------------------

--------------\*\*\*\*\*\*\*\*Union Table --union remove duplicate

-------------------union all does not remove duplicate

--\*\*\*\*\*\*\*\*\* importent thing is colun shoul match and data types should match

select department from employees

union

select department from departments;

select department from employees

union all

select department from departments;

select distinct department from employees

union all

select department from departments;

--\*\*\* order by should be after last table in union

--because union is consider in the tables

select distinct department from employees

union all

select department from departments

order by department;

select distinct department from employees

union all

select department from departments

union

select country from regions ;--union give just one column

-------------------EXCEPT: take the first result set and remove

--all the data which i sfound in the second result

select distinct department from employees

except

select department from departments;--very cool as we can simply see what is inside one which is not another one

--pervious we used sub quaries

select department, count(\*) from employees

group by department

union all

select 'total' , count(\*) from employees

-----------------------\*\* CartJES PRODUCTS --CROSS JOINT

--selecting from multiple table without joining them

--the result is a table with multiplied rows of tables

--each record is matched with following table

---\*\*\*\*\*\*\*\*\*excercise

select first\_name, department, hire\_date, country

from employees a

left join

regions b

on a.region\_id=b.region\_id

where hire\_date in ((select max(hire\_date)

from employees), (select min(hire\_date)

from employees))

(select first\_name, department, hire\_date, country

from employees a

left join

regions b

on a.region\_id=b.region\_id

where hire\_date in (select min(hire\_date)

from employees)

limit 1)

union

select first\_name, department, hire\_date, country

from employees a

left join

regions b

on a.region\_id=b.region\_id

where hire\_date in (select max(hire\_date)

from employees)

---------moving range quary\*\*\*\*\*\*\*\*\*\*

select hire\_date, salary, (select sum(salary) as salary\_pattern from employees e2

where hire\_date between e.hire\_date-90 and e.hire\_date)

from employees e

order by hire\_date

---------view we can not manupulated data in view

create view v\_employeeiformation as

select hire\_date, salary, (select sum(salary) as salary\_pattern from employees e2

where hire\_date between e.hire\_date-90 and e.hire\_date)

from employees e

order by hire\_date

---view is not a table we can just pull of data from it

------\*\*\*\*\*inline view is for subquaries

------------------------------\*\*\*window

--instead of using corelated subquaries which

-- should be run for each record which is expensive

--we can use window function

--which is nothing but

--\* agggregation function +over( partition by column name)

select first\_name, department,(select count(\*) from employees e2

where e2.department=e1.department)

from employees e1

order by department

select first\_name, department,

count(\*) over(partition by department)

from employees;

select first\_name, department,

sum(salary) over(partition by department),

region\_id,

count(\*) over(partition by department)

from employees;--first from then where and then select part will be run

--- windows

select first\_name ,department, (select count(\*) from employees where department = e.department)

from employees e;

select first\_name ,department, count(\*) over (partition by department )

from employees e;

--window change on the basis of partition

--framing

--window is a group of data that we compute

--an aggrigate on

--order data that allowes

--you for other type of computation

-- running total is meaning full when it is

--on the basis of order of time or other things

-- structuring the window in an ordered way

--frame is coming when we considering a rang by this statment:

--"order by ... range between unbounded preceding and current row

"

select first\_name, hire\_date,

sum(salary) over( order by hire\_date range between

unbounded preceding and current row) as running\_total\_of\_salaries

from employees;-- sql as a defult has the range to current row

--when we use order by in the over clause

-- for the above example we can have

select first\_name, hire\_date,

sum(salary) over( order by hire\_date ) as running\_total\_of\_salaries

from employees;-- with the same exact result

-- we can add partition and runnig total computed on each partion or group seperately

select first\_name, hire\_date, salary,

sum(salary) over( partition by department order by hire\_date ) as running\_total\_of\_salaries

from employees;

-- adjacent salary دنباله فیبونانچی

select first\_name, hire\_date,

sum(salary) over( order by hire\_date rows between

1 preceding and current row) as running\_total\_of\_salaries

from employees;

-- self study rows followig and preceding , overall, window functions and framing

--- window functions \*\*\*Rank function

--rank data based on some criteria

select first\_name, email, department, salary,

rank() over(partition by department order by salary desc)

from employees;

--now if we want see the ranked 8 employees

/\*select first\_name, email, department, salary,

rank() over(partition by department order by salary desc)

from employees

having rank =8;--or where caluse dose not work as windo will execute

--after where or having clause so we need subquary\*/

select \* from(select first\_name, email, department, salary,

rank() over(partition by department order by salary desc)

from employees)a

where rank=8;

-- what if we want to rank group of employees ?

--make a bracket or دسته بندی کردن

select first\_name, email, department, salary,

ntile(5) over(partition by department order by salary desc)

from employees

-- takes a column as an argument>>FIRST\_VALUE() function

--takes the first value of the window

--and repeated for the entire rows

select first\_name, email, department, salary,

first\_value(salary) over(partition by department order by first\_name desc)

from employees;-- we havethe same for the Nth\_VAlue\*\*\*\*\*\*\*

select first\_name, email, department, salary,

nth\_value(salary,5) over(partition by department order by first\_name desc)

from employees;

-- lead and lag function ,

-- allow us to work with other records

--reletive to the current records, having the next or the prior record

select first\_name, email, department, salary,

lead(salary) over()

from employees;

select first\_name, email, department, salary,

lag(salary) over()

from employees;

-- next higher salary column / closest higer paid employee

select first\_name, email, department, salary,

lag(salary) over(partition by department order by salary desc)

from employees;--

--closest lower paid employee

select first\_name, email, department, salary,

lead(salary) over(partition by department order by salary desc)

from employees;

--\*\*\*\*\*\*\*Rollups and Cubes--work with group by clause

--

CREATE TABLE sales

(

continent varchar(20),

country varchar(20),

city varchar(20),

units\_sold integer

);

INSERT INTO sales VALUES ('North America', 'Canada', 'Toronto', 10000);

INSERT INTO sales VALUES ('North America', 'Canada', 'Montreal', 5000);

INSERT INTO sales VALUES ('North America', 'Canada', 'Vancouver', 15000);

INSERT INTO sales VALUES ('Asia', 'China', 'Hong Kong', 7000);

INSERT INTO sales VALUES ('Asia', 'China', 'Shanghai', 3000);

INSERT INTO sales VALUES ('Asia', 'Japan', 'Tokyo', 5000);

INSERT INTO sales VALUES ('Europe', 'UK', 'London', 6000);

INSERT INTO sales VALUES ('Europe', 'UK', 'Manchester', 12000);

INSERT INTO sales VALUES ('Europe', 'France', 'Paris', 5000);

--like a grid

--having breakdown in one quary

select continent, country , city, sum(units\_sold)

from sales

group by grouping sets(continent, country , city)

select continent, country , city, sum(units\_sold)

from sales

group by grouping sets(continent, country , city,()--nothing)

--grouping with combinations considering the combination from the right

--first three continent, country , city

--then continent, country

--then continent

select continent, country , city, sum(units\_sold)

from sales

group by rollup(continent, country , city);

--to have all of the combinations we need to use CUBES

select continent, country , city, sum(units\_sold)

from sales

group by cube(continent, country , city);