select \* from customer\_table ;

alter table customer\_table add test varchar(255)

alter table customer\_table drop test ;

alter table customer\_table drop column test ;

alter table customer\_table alter column age type varchar(255) ;

alter table customer\_table rename column first\_nam to first\_name ;

alter table customer\_table alter column cust\_id set not null ;

insert into customer\_table(first\_name, last\_name, age, email\_id) values

('aa', 'bb', '25' , 'abc@xyz.com') ;

alter table customer\_table alter column cust\_id drop not null ;

DELETE FROM customer\_table WHERE cust\_id IS NULL;

alter table customer\_table add constraint cust\_id check (cust\_id>0) ;

alter table customer\_table add primary key (cust\_id) ;

UPDATE Science\_class SET science\_marks = 45 WHERE enrollment\_no = 1;

select \* from customer ;

select \* from product ;

select \* from sales ;

select \* from customer where city in ('Philadelphia', 'Seattle') ;

select \* from customer where city = 'Philadelphia' OR city ='Seattle' ;

select \* from customer where age between 20 and 30 ;

select \* from customer where age >= 20 and age <= 30 ;

select \* from customer where age not between 20 and 30 ;

/\*

multi line comments

\*/

--single line comments

/\* Like Comments\*/

/\*

Here % and \_ is called as wild card

% represents n number of charachters

\_ represents one charachter

\*/

select \* from customer where customer\_name like 'J%';

select \* from customer where customer\_name like '%Nelson%';

select \* from customer where customer\_name like '\_\_\_\_ %';

/\*customer has 4 words in firtst name and with 'n' number of words as second name \*/

select \* from customer where city not like 'S%';

select \* from customer where customer\_name like 'G\%';

/\*Here the \ is know as the escape charachter which treats % as a charachter and not as wild card\*/

/\*exercise\*/

select \* from customer order by customer\_name asc ;

select distinct city from customer where region in ('Central', 'East');

select \* from sales where sales between 100 and 500 ;

select distinct customer\_name from customer where customer\_name like '% \_\_\_\_';

/\* oder by \*/

select \* from customer where state = 'California' order by customer\_name;

select \* from customer where state = 'California' order by customer\_name asc;

select \* from customer where state = 'California' order by customer\_name desc;

select \* from customer order by city asc, customer\_name desc;

select \* from customer where state = 'California' order by city asc, customer\_name desc;

/\*

ORDER BY 2 DESC;

where 2 indicates the column number without specifying te name of the column

\*/

select \* from customer order by 2 asc;

select \* from customer order by age desc;

/\*Limiting number of return outputs\*/

select \* from customer where age >= 25 order by age desc limit 8;

select \* from customer where age > 25 order by age asc limit 10;

/\* exercise \*/

select \* from sales limit 5 ;

select \* from sales where discount > 0 order by discount desc ;

select \* from sales where discount > 0 order by discount desc limit 10;

/\* AS alias

provides second name for the column name or table name

\*/

select customer\_id as "Serial Number", customer\_name as "Name", age as "Customer\_age" from customer ;

select customer\_id as "Serial Number", customer\_name as Name, age as Customer\_age from customer ;

/\* "" is used for names with spaces or to retain the capitial initials

if not used then the names are named as small letters like Name will be changed to name

run this querry for better understanding

\*/

/\* COUNT \*/

select count(\*) from sales ;

select count(order\_line) as "Number Of Products Ordered", count (distinct order\_id) as "Number of Orders" from sales where customer\_id = 'CG-12520';

/\* SUM \*/

select sum(profit) as "Total Profit" from sales ;

select sum(quantity) as "Total Quantity" from sales where product\_id = 'FUR-TA-10000577' ;

/\* AVERAGE \*/

select avg(age) as "Average Customer Age" from customer ;

select avg(sales \* 0.10) as "Average Commision Value" from sales ;

/\* MIN MAX \*/

select min(sales) as "Minimum Sales Value June" from sales where order\_date between '2015-06-01' and '2015-06-30' ;

select sales from sales where order\_date between '2015-06-01' and '2015-06-30' order by sales asc ;

select max(sales) as "Maximun Sales Value June" from sales where order\_date between '2015-06-01' and '2015-06-30' ;

/\* exercise \*/

select sum(sales) as "Total Sales" from sales ;

select count(distinct customer\_id) from customer where age between 20 and 30 ;

select avg(age) as "Average Age of Customers In East Region" from customer where region in ('East') ;

select min(age) as "Minimum Age of Customer", max(age) as "Maximum Age Of Customer" from customer where city in ('Philadelphia');

select min(age) as "Minimum Age of Customer", max(age) as "Maximum Age Of Customer" from customer where city like ('P%a');

/\* GROUP BY \*/

select region, count(customer\_id) as "Customer Count" from customer group by region ;

select product\_id, sum(quantity) as "Quantity Sold" from sales group by product\_id order by "Quantity Sold" desc ;

/\* ALL QUEREY LEARNED SO FAR \*/

select customer\_id, min(sales) as "Minimun Sales", max(sales) as "Maximun Sales", avg(sales) as "Average Sales", sum(sales) as "Total Sales" from sales group by customer\_id order by "Total Sales" desc limit 5 ;

/\* HAVING \*/

select region, count(customer\_id) as "Customer Count" from customer group by region having count(customer\_id)>200;

select region, count(customer\_id) as "Customer Count" from customer where customer\_name like 'A%' group by region ;

select region, count(customer\_id) as "Customer Count" from customer where customer\_name like 'A%' group by region having count(customer\_id) >15;

/\* exercise \*/

select \* from sales limit 1 ;

select \* from customer limit 1 ;

select sum(sales) as "Total sales", sum(quantity) as "Total quantity", count(order\_id) as "Number of Orders", max(sales) as "Max Sales Value", min(sales) as "Min Sales Value" , avg(sales) as "Average Sales Value" from sales ;

select product\_id, count(product\_id) as "List Of Product IDs" from sales group by product\_id having count(quantity) > 10 ;

/\* CASE EXPRESSIONS \*/

SELECT \*,

CASE WHEN age < 30 THEN 'Young'

WHEN age > 60 THEN 'Citizen'

ELSE 'Middle Aged'

END AS Age\_Category

FROM customer ;

/\* JOINS \*/

/\*Creating sales table of year 2015 \*/

/\* Creating table with customre age between 20 and 30 \*/

/\*Creating sales table of year 2015\*/

Create table sales\_2015 as select \* from sales where ship\_date between '2015-01-01' and '2015-12-31';

select count(\*) from sales\_2015; --2131

select count(distinct customer\_id) from sales\_2015;--578

/\* Customers with age between 20 and 60 \*/

create table customer\_20\_60 as select \* from customer where age between 20 and 60;

select count (\*) from customer\_20\_60;--597

/\* INNER JOIN || (A n B)

Gives the Intersection of two tables

\*/

select

a.order\_line,

a.product\_id,

a.customer\_id,

a.sales,

b.customer\_name,

b.age

from sales\_2015 as a

inner join customer\_20\_60 as b

on a.customer\_id = b.customer\_id

order by customer\_id ;

/\* LEFT JOIN || (A U B')

outputs all of A and intersection of A and B

\*/

select

a.order\_line,

a.product\_id,

a.customer\_id,

a.sales,

b.customer\_name,

b.age

from sales\_2015 as a

left join customer\_20\_60 as b

on a.customer\_id = b.customer\_id

order by customer\_id ;

/\* RIGHT JOIN || (A' U B)

outputs all of B and intersection of A and B

to get all the values of right table, always remember to select the common coulmn

name from the right table rather than from the left table in right join

\*/

select

a.order\_line,

a.product\_id,

b.customer\_id, --selecting the common coulmn from the right join table

a.sales,

b.customer\_name,

b.age

from sales\_2015 as a

right join customer\_20\_60 as b

on a.customer\_id = b.customer\_id

order by customer\_id ;

/\* FULL JOIN || (A U B)

Like union join but different

It adds the extra colum and null rows to the output table

And full join could be performed only when there is atleast one relation between the two or more tables

\*/

select

a.order\_line,

a.product\_id,

a.customer\_id,

a.sales,

b.customer\_name,

b.age,

b.customer\_id

from sales\_2015 as a

full join customer\_20\_60 as b

on a.customer\_id = b.customer\_id

order by a.customer\_id, b.customer\_id ;

/\* CROSS JOIN

creates cartesian product between two sets of data

\*/

create table month\_values (MM integer) ;

create table year\_values (YYYY integer) ;

insert into month\_values values (1), (2), (3),(4), (5), (6),(7), (8), (9), (10), (11),(12);

insert into year\_values values (2011), (2012), (2013),(2014), (2015), (2016),(2017), (2018), (2019);

select \* from month\_values ;

select \* from year\_values ;

select a.YYYY , b.MM

from year\_values as a, month\_values as b

order by a.YYYY, b.MM ;

/\* EXCEPT || (A n B')

output contains values of A exculded of the values common to A and B

\*/

select customer\_id

from sales\_2015

except select customer\_id from customer\_20\_60

order by customer\_id ;

/\* UNION || (A U B)

The union adds the extra rows in the output if there is a common coulumn in both or relse it adds the coulum in output

It avoids the null data in the row or column I guess

Union can be used for 2 or more tables with or without common links between them

\*/

select customer\_id

from sales\_2015

union select customer\_id from customer\_20\_60

order by customer\_id;

/\* exercise \*/

select \* from sales \_2015 limit 1;

select \* from customer\_20\_60 limit 1;

select b.state , sum(sales) as total\_sales

from sales\_2015 as a left join customer\_20\_60 as b

on

a.customer\_id = b.customer\_id

group by

b.state ;

select \* from sales limit 1;

select \* from product;

select

a.\*, sum( b.sales ) as total\_sales , sum(quantity) as total\_quantity

from product as a left join sales as b

on

a.product\_id = b.product\_id

group by

a.product\_id

/\* SUBQUERY \*/

select \* from sales where customer\_id in (select customer\_id from customer where age > 60) ;

select

a.product\_id,

a.product\_name,

a.category,

b.quantity

from product as a

left join (select product\_id, sum(quantity) as quantity from sales group by product\_id) as b

on a.product\_id = b.product\_id

order by b.quantity desc ;

select customer\_id, order\_line,

(select customer\_name from customer where customer.customer\_id = sales.customer\_id)

from sales

order by customer\_id ;

/\* notes on subquery

subquery takes more time and resourses from database than joins

subquery must be enclosed within paranthesis

ulinke 'in' command, 'between' command cannot be used between query and subquery but

it can be used within subquery

\*/

/\* exercise \*/

select

a.\*,

b.customer\_name,

b.age,

b.product\_name,

b.category

from sales as a

left join (select

c.customer\_name,

c.customer\_id,

c.age,

d.product\_name,

d.category

from customer as c

full join product as d

on c.customer\_id = d.product\_id ) as b

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

select

c.customer\_name , c.age , sp.\* from

customer as c

right join (select s.\*,

p.product\_name , p.category

from sales as s

left join product as p

on

s.product\_id = p.product\_id ) as sp

on

c.customer\_id = sp.customer\_id

/\* VIEWS

used to control the data necessary for the team to visible

instead of creating a new table we are creating an instance of a table (i guess)

\*/

create view logistics as

select a.order\_line,a.order\_id,b.customer\_name,b.city,b.state,b.country

from sales as a

left join customer as b

on a.customer\_id = b.customer\_id

order by a.order\_line ;

select \* from logistics ;

/\* create or replace view logistics as

this commandis best to use because in case if the instance of table has already

been created this command is used to update the instance as we need it

\*/

drop view logistics ; --delete the view

--we can also update a view with new values but is not adviseable

/\* INDEX \*/

/\* Single column index is called as simple index

more than 1 coulmn index is known as composite index

\*/

create index mon\_idx

on month\_values(MM);

/\* Droping Index\*/

/\* drop index 'if exist command' is used to check wheather the index is created or not

if created it deletes the index else does not throw an error

advisable to use for droping anything (tables)

\*/

/\* CASCADE it deletes the dependable database object that relies on this index

\*/

/\* RESTICT throws an error if that index is used by other database

\*/

drop index mon\_idx ;

/\* ALTER INDEX is used to rename a index

Syntax : Alter index [if exists] index\_name, rename to new\_index\_name;

\*/

/\* good practices : it is best to create an index with column that has an integer type

because it requires less space compared to other index

\*/

/\* Exercise \*/

select \* from sales order by order\_date desc limit 5 ;

create view Daily\_Billing as select order\_line,product\_id,sales,discount from sales where order\_date = '2017-12-30' ;

select \* from Daily\_Billing ;

drop view if exists Daily\_Billing ;

create or replace view Daily\_Billing as select order\_line , order\_date, product\_id , sales, discount

from sales where order\_date in (select max( order\_date ) from sales ) ;

/\* LENGTH (STRING) \*/