SELECT STATEMENT(Jithin)

RDBMS(relational database management system)
INFORMATION IS LINKED FROM DIFFERENT TABLES

- Select * from demo
- SELECT name,id from demo(column selection)
- select * from demo **LIMIT** 3
- select count(*) from demo
- select * from demo order by name
- select * from demo order by ID desc
- select * from demo order by id desc limit 2
- insert into demo (id,name,hint) values (7,'test',2)
- insert into demo values (9,'test',4)
- insert into demo values (10,'test',2)
- select DISTINCT name,hint from demo
- select DISTINCT id from demo order by id
- select DISTINCT id from demo order by id DESC
- select name as Fulll_name,hint as temp_name from demo limit 5
- select name as "Full Name",hint as "temp Hint" from demo limit 5
- select name "full name" from demo
- insert into demo values(12,'test2',14)

Select with where statement

- select * from demo where name not **like** '%test%' or id>5
- select * from demo where name not like '%test%' and name not like'%limit%' and id<5
- select min(id) asmin_id,max(id)as max_id,sum(id)as sum_id,avg(id) as avg_id,count(id) as count_id from demo
- select min(id) asmin_id,max(id)as max_id,sum(id)as sum_id,round(avg(id),2) as avg_id,count(id) as count_id from demo

Sales data summation using (group By must be with AGGREGATION)

- select deal_num,amount from deals
- select deal_num,sum(amount) as total_amount from deals group by deal_num
- SELECT DISTINCT(not mandat name as 'name',sum(amount) as 'amount',count(product)as 'number of deals by product' from table from deals **group b**y name;

SQL JOINTS

Union- just append the data add table at the bottom

Joint types

- ❖ Inner join- get only common fields

- Left joint -all left available(mostly used)
- Right joint- all right is available
- Outer join-all the rows are combined
- Cross join- all permutation combination

Inner joint

- Multiple keys

```
Select d.deal_num,d.sales_per, dim.country
 from deals as d
 Inner joint dimCountry as dim
On d.country_code = dim.country_code
And d.random_col=dim.random_col;
```

- Multiple joints

(1+2)+3- two joints

```
Select d.deal_num,d.sales_per, dim.country,dc.qtr_num
 from deals as d
 Inner joint dimCountry as dim
On d.country_code = dim.country_code
And d.random_col=dim.random_col
Left outer join dim_calender as dc
On d.date=dc.calender_data;
FULL outer join dim_sales as ds
On d.date=dc.calender_data;
```

sub-Queries

1. From
2. Joints
3. Where

```
select * from deals
where date=(
select max(date)from deals)
```

Practice

- **Query to extract top 5 deals by product**

```
selected * from deals  order by amount desc limit 5
```

- **Query to extract deal and total amount in 2020 and qtr_num=2**

```
 select d.deal_num as 'deal',sum(d.amount) as 'total_amount' from deals as d
inner joint dimCalender as dc
on   d.date=dc.calender_date
```

```
            where dc.qtr_num= 2 and dc.Year_num=2020 order by deal_num
```

- **Query to extract deal and total amount**

```
select deal_num as 'deal',sum(deal) as 'total_amount' from deals  order by deal_num
```

- **Query to extract all deals data on latest date**

```
select * from deals where date=(
select max(date)from deals
 )
```

- **Query to extract all deals data on yesterday date**

```
select * from deals where  date<>(
select  max(date) from deals)
select max(date)from deals
)
```

Khan academy SQL

Practice :-https://sqliteonline.com/

```
/**CREATE TABLE groceries(id INTEGER PRIMARY KEY ,name Text,quantity INTEGER);

INSERT INTO groceries VALUES(1,"bananas",4);
INSERT INTO groceries VALUES(2,"oranges",2);
INSERT INTO groceries VALUES(3,"bananas",3);
**/
select * from groceries
```

```
ALTER TABLE groceries ADD COLUMN aisle INTEGER;

INSERT INTO groceries VALUES(4,"bananas",7,6);
INSERT INTO groceries VALUES(5,"oranges",1,8);
INSERT INTO groceries VALUES(6,"bananas",2,10);

INSERT INTO groceries VALUES(7,"mango",7,6);
INSERT INTO groceries VALUES(8,"oranges",1,8);
INSERT INTO groceries VALUES(9,"berries",2,10);
INSERT INTO groceries VALUES(10,"mango",7,6);
INSERT INTO groceries VALUES(12,"pineapple",1,8);
INSERT INTO groceries VALUES(11,"coconut",2,10);
```

```
select * from groceries  ORDER BY aisle ;
**/
select * from groceries where aisle<8  ORDER BY aisle ;
```

```
/**create TABLE market(id integer primary key,name text,cat text,stock integer,aisle integer);
INSERT INTO  market values(1,"mixer","electronics",10,25);
INSERT INTO  market values(2,"grinder","electronics",15,25);
INSERT INTO  market values(3,"fridge","electronics",3,25);
INSERT INTO  market values(4,"stove","electronics",7,25);
INSERT INTO  market values(11,"tv","electronics",1,2);
INSERT INTO  market values(12,"laptop","electronics",4,2);
INSERT INTO  market values(13,"mobile","electronics",23,2);
INSERT INTO  market values(14,"charger","electronics",17,2);
INSERT INTO  market values(111,"basket","goods",1,21);
INSERT INTO  market values(112,"bed lamp","goods",4,21);
INSERT INTO  market values(113,"table","goods",23,22);
INSERT INTO  market values(114,"sofa","goods",17,22);
SELECT * from market;
select aisle,cat,max(stock) FROM market ORDER by cat;
select aisle,cat,max(stock) FROM market GROUP by cat;

select name,cat from market WHERE aisle>10 and stock<15 ;
**/
select loan,sum(price) from showroom  where price<2876 group by LOAN;
select loan,cc,year from showroom  where price<2876 order by LOAN;
```

```
 SELECT * FROM artists;
SELECT * FROM songs;

SELECT title  FROM songs WHERE artist ="Queen";
SELECT name  FROM artists WHERE genre ="Pop";

SELECT title  FROM songs WHERE artist IN (SELECT name FROM artists where genre not
in ("Pop","Country");
**/
sELECT name FROM artists where genre in ("Pop");
```

```
SELECT name,number_grade,round(100*fraction_completed)as percent_completed FROM
student_grades;

SELECT name,number_grade,
CASE
```

```
  WHEN number_grade >90 THEN "A"
  WHEN number_grade>80 THEN "B"
  WHEN number_grade>70 THEN "C"
  ELSE "F"
  END AS letter_grade
FROM student_grades;

SELECT COUNT(*),
CASE
  WHEN number_grade >90 THEN "A"
  WHEN number_grade>80 THEN "B"
  WHEN number_grade>70 THEN "C"
  ELSE "F"
  END AS letter_grade
FROM student_grades GROUP BY LETTER_GRADE;
```

```
SELECT name,MIN(population) as minimum FROM countries;
SELECT name,max(population) as  maximum FROM countries;
SELECT avg(population) as Average  FROM countries;

SELECT name,avg(population) as avg_popu from countries group by name having
avg_popu>1000000000;

SELECT count(*)as countries_number,
CASE
when population>100000000 then "HP"
when population >10000000 then "AP"
when population>100000 then "MP"
else "LP"
end as Range_pop
from countries group by Range_pop;
select * from countries where (density_per_sq_km >200 and area_sq_km >500000) or
(fertility_rate>5 and median_age>20);
```

```
insert into persons (name,age) VALUES("rajesh",50);
insert into hobbies (person_id,name) VALUES(5,"rowing");

 select hobbies.name,persons.name from persons
  join hobbies
  on persons.id=hobbies.person_id;

   select hobbies.name,persons.name from persons
  join hobbies
  on persons.id=hobbies.person_id
  where persons.name="Bobby McBobbyFace"
```

```
SELECT * from orders;

select customers.name,customers.email,orders.item,orders.price
 from customers
 join orders
 on customers.id=orders.customer_id;

select customers.name,customers.email,orders.item,orders.price
 from customers
 left OUTER join orders
 on customers.id=orders.customer_id ;
  **/
select customers.name,customers.email,sum(orders.price) as total_amount
 from customers
 left OUTER join orders
 on customers.id=orders.customer_id  GROUP by customers.name order by total_amount
desc;
self-join
select movies.title,sequel.title
FROM movies
join movies sequel
on movies.sequel_id=sequel.id;

select movies.title,sequel.title
FROM movies
left outer join movies sequel
on movies.sequel_id=sequel.id;

 select persons.fullname,a.fullname from friends
   join persons
   on friends.person1_id=persons.id
   join persons a
   on  friends.person2_id=a.id;

UPDATE and DELETE
update documents set author= "jackie Draper"
WHERE author="Jackie Paper";
  select * from documents;
  DELETE  from documents where title like "%Things I'm Afraid%";
  select * from documents;
```