Date: _____
Page: _____

SQL Command Categories

(i) Data Definition language (DDL)

→ Consists of command that can be used to define the database schema.

→ CREATE, DROP, ALTER, TRUNCATE.

(ii) Data manipulation language (DML)

→ The SQL Command that deals with manipulation of data present in database.

→ SELECT, INSERT, UPDATE, DELETE.

(iii) Data Control language (DCL)

→ Includes the Command that mainly deals with right permission and other controls of the database system.

→ GRANT, INVOKE.

Date: .....
Page: .....

(iv) Transaction control language

→ Includes the commands which mainly deals with the transaction of database.

→ COMMIT, ROLLBACK, SAVEPOINT.

DELETE → used to delete tuple in a database
→ It generate log file
→ Roll back is possible.

DROP → used to delete complete structure of schema.

TRUNCATE → used to delete all records at a time.
→ NO log file.
→ NO Rollback.

Date:
Page:

Constraint in Database

(I) NOT NULL
→ Ensures that a NULL value can't be stored in a column.

(II) UNIQUE
→ This constraint make sure that all the values in a column are different.

(III) Check
→ This constraint ensure that all the values in a column satisfy a specific condition.

(IV) DEFAULT
→ This constraint consists of a set of default values for a column when no value is specified.

(V) INDEX
→ ~~A technique~~ This constraint is used to create and retrieve data from the database very quickly.

Date : .....
Page : .....

Postgres cmd

(I) psql -U postgres -h localhost
↳ for starting Connection.

(II) psql -U postgres -h localhost -P 5432 student
↳ to connect student database

(III) \c student
→ If you are in postgres. then to Connect student database.

(IV) \q
→ To quit.

(V) \l
→ To show the database.

(VI) \du
↳ To show the role and database.

(VII) \d
↳ for describe

(VIII) \d person

Date: _____
Page: _____

(IX) \i <loc_of_file\file_name.sql>
↳ to execute .sql file.

(X) \l database or \dt
→ to show the tables.

(XI) \df
→ list of function.

Date: .....
Page: .....

CREATE TABLE

```
Create table person (
    id BIGSERIAL NOT NULL PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    gender VARCHAR(5) NOT NULL,
    date_of_birth DATE NOT NULL
    );
```

INSERT VALUES

```
INSERT INTO person (
        first_name,
        last_name,
        gender,
        date_of_birth ) VALUES
('test', 'data', 'Female', '1990-08-11');
```

Remove Duplicate

```
Select distinct (country) from person.
```

↳ unique country from person table.

Date:
Page:

Comparision operator

Select 1=1;
o/p = t (True)

Select 1=2;
o/p = f (False)

Select 3<>4;
o/p = t (True)

LIMIT OFFSET AND FETCH

LIMIT

Select * from person where gender = 'male' limit 10;

↳ first 10 record with gender male.

OFFSET

Select * from person offset 5 limit 10;

↳ [At [5:15]

starting record - 1

no. of records

FETCH

↳ similar to limit.

```
select * from person offset 5 fetch
        first 10 row only;
```

IN

```
select * from person where
country IN ('china', 'nepal');
```

= select * from person where country = 'china' or country = 'nepal'.

BETWEEN

```
select * from person where date_of_birth
BETWEEN '2021-01-01' AND
        '2021-05-05';
```

LIKE

```
select * FRom person where
email Like '%.uk';
```

Date:
Page

Like operator	Description
'a%'	→ finds any value that starts with "a".
'%a'	→ finds any value that end with a.
'%or%'	→ finds any value that have "or" in any position
'_r%'	→ finds any value that have 'r' in second position.
'a__%'	→ finds any values that start with "a" and are at least 3 character in length.
'a%o'	→ find any values that starts with 'a' and end with 'o'.

GROUP BY

```
select country, count(*) from person group
                                BY country;
```

country	count
nepal	5
china	3
USA	7

Date: .....
Page: .....

HAVING

The HAVING clause was added to SQL because where keyword can't be used with aggregate functions.

Syntax:-

```
Select    column_name(s)
from      table_name
WHERE     condition
GROUP BY  column_name(s)
HAVING    condition
ORDER BY  column_name(s);
```

MIN, MAX & AVG (agg function)

MIN

```
Select min(age) from person;
```

MAX

```
Select max(age) from person;
```

Sum

```
Select sum(price) from car;
```

Date:
Page:

AVG

select ROUND(AVG(price)) from car;

Arithmetic operator & Alias

arithmetic operator

select id, make, price, model, price * .10 "discounted price" from car;

Alias

COALESCE

↳ similar to default.

select COALESCE(null, 1, 10) AS number;

o/p 1

Select coalesce (email, 'not available') from person;

Date :
Page :

Date and Timestamps

```
Select now();
```

```
2021-12-03 22:43:41.77
```

```
Select now()::DATE;
```

```
2021-12-03
```

Adding & subtracting Dates

```
Select now()::DATE + INTERVAL '1 YEARS';
```

Extracting fields from dates

```
Select extract (year from now());
```

```
2021
```

Date:
Page:

Age function

```
select first_name, last_name, gender, country,
Age(now(), date_of_birth) as age
from person;
);
```

DELETING RECORDS

```
Delete from person where id = 1001;
```

updating Records

```
update table_name set column1=value1
column2=value2 where condition;
```

ON CONFLICT DO NOTHING

```
INSERT into person (id, first_name, last_name,
email, gender, date_of_birth, country) values(
2023, 'Latrena', 'Dirt', 'test@.com', 'male',
'2021-11-27', 'portugal') ON CONFLICT
(email) DO NOTHING;
);
```

if INSERT 0 0

Date: .....
Page: .....

ON CONFLICT DO UPDATE

```
Insert into person (id, first_name,
last_name, gender, email, date_of_birth,
country) values (2017, 'Russ', 'Ruddock',
'male', 'rr@12.com' Date '1952-02-3',
'Norway') ON CONFLICT (id) DO
UPDATE set email = excluded.email;
```

op INSERT 0 1

SEQUENCES

```
Select * from person_id_seq;
```

op

last_value	log_cnt	is_called
3	30	t

```
Select nextval('person_id_seq'::reg class);
```

op next val
1

Date: _____
Page: _____

```
ALTER SEQUENCE person_id_seq
                    RESTART 10;
```

EXTENSIONS

```
Select * FROM pg_available_extension;
```

UUID (Universally unique Identifier)

```
Create extension if not exists
                    "uuid-ossp";
Select uuid_generate_v4();
# 29598ee8-6b30-404e-b1e6
```

UUID IN ACTION

```
create table Car (
    Car_uid UUID not null primary key,
    - - -
);
```

Date: _____
Page: _____

```
insert into car (car_uid, make,
model, price) values(uuid_generate_v4(),
'land Rover', 'sterling', '87865.38');
```

EXPORTING TO CSV

```
\copy (select * from person left join
    car ON person.car_uid = car.car_uid)
TO '/home/bibek/Desktop/MySQL/result'
DELIMITER ',' CSV HEADER;
```

off copy ;

copy table All data

```
create table new_table
AS
TABLE existing_table;
-- WHERE
-- condition;
```

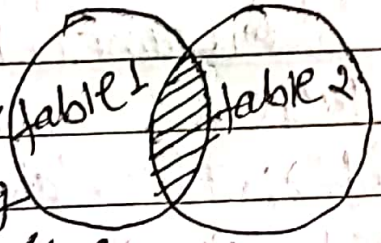
Date: _____
Page: _____

// Copy only structure:

```
create table new_table
AS
Table existing_table
with NO Data;
```

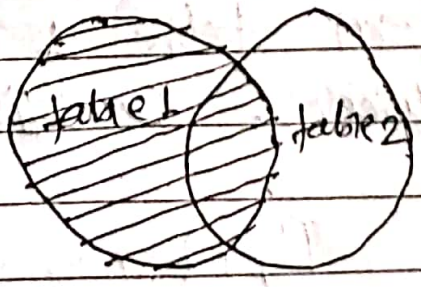
INNER JOIN (Default Join)

↳ The Inner join keyword selects records that have matching values in both tables.




```
Select column_name(s)
from table1
Inner Join table2
ON table1.column_name = table2.column_name;
```

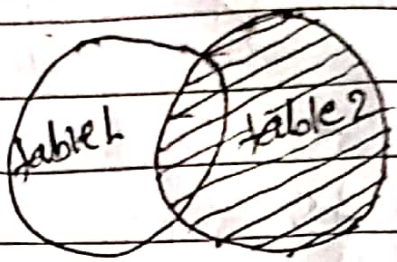
LEFT JOIN




```
Select column_name(s)
from table1
Left Join table2
ON table1.column_name = table2.column_name;
```

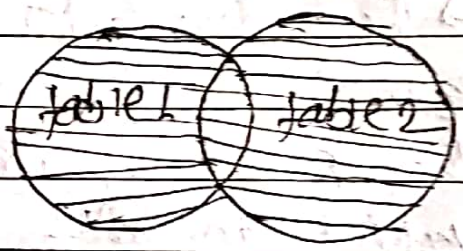
Date: _____
Page: _____

Right Join




```
Select column_name(s)
from table1
Right Join table2
on table1.column_name = table2.column_name;
```

Full / Full outer Join




```
Select column_name(s)
from table1
full outer Join table2
on table1.column_nam=
table2.column_name
WHERE condition;
```

Self Join

```
Select column_name(s)
from table T1, table T2
where condition;
```

T1, T2 are diff. tables aliases for the same table.

Date:
Page:

I. m. P

```
Select book_name, price||'$' "price in $"
                        from library.
```

O/P

book_name	price in '$'
A	80$
B	20$
C	5 $

Primary Key

↳ primary keys must contain unique values, and can't contain NULL values.

↳ A table can have only one primary key and in the table, this primary key can consists of single or multiple columns fields.

```
create table person (
        id int not null,
        _ _ _
constraint PK_person primary key(id)
        );
```

Date: _____

Page: _____

Foreign Key

A foreign key is a field in one table, that refers to the primary key in another table.

A table with a foreign key is called a child table. table with primary key is called referenced or parent table.

```
create table orders (
    order_id int not null,
    primary key (order_id),
    constraint fk_orders foreign
    key (person_id) references
                person (person_id)
    );
```

Date:
Page:

ALTER TABLE

Alter table table_name [action--]

action is:
ADD [column]
DROP [column]
Alter [column] [set | Drop] default
Alter [column] [set | Drop] not null
ADD constraint

(I) add a column.
↳ Alter table orders ADD column
vendor_name varchar (25);

(II) change column datatype:
↳ Alter table orders Alter column col_name
Type varchar (25);

(III) Rename column
↳ Alter table orders rename column
city to city_ven.

Date:
Page:

(iv) Add NOT NULL Constraint

↳ Alter table orders Alter Column City set NOT NULL;

Note: primary key, foreign key, unique & check have same syntax.

Alter table orders add Constraint unique_id unique(id);

Alter table orders add Constraint pk_order primary key(id);

Update:

↳ update command is used to modify the existing data.

update table_name Set Column_name1 = new_value1, Column_name2 = new_value2 WHERE Condition;

Date:
Page:

DELETE

DELETE from book WHERE price < 25;

Difference Between

(1) WHERE & HAVINGS

a) WHERE
→ filter rows
→ works on row's data, not on aggregated data.

Select * from emp where score >= 40;

b) HAVINGS
→ works on aggregated data.

Select MAX(salary) from employee;

Aggregate function
→ to perform calculations on multiple rows of a single column.
→ It returns a single value
→ It is used to summarize data.
→ Count, max, min, AVG, sum.
→ Having, group By.

Date: _____
Page: _____

e.g.

```
Select max(salary) from Emp
Having emp_id 7105;
```

(11) UNION Vs UNION ALL

union → removes duplicate records
union All → doesn't remove duplicate

```
Select city from customers
union
select city from suppliers
```

o/p
→ unique city name

```
select city from customers
union All
Select city from suppliers
```

→ union operator combines result set of 2 or more select statements.

Date:
Page:

→ each select statement must have
(i) same number of columns
(ii) columns must have similar datatype.
(iii) columns must be in same order.

(11) IN Vs EXISTS

a) IN
→ multiple OR

```
select * from customers where
city in (mumbai, Bangalury, chennai);

select * from customers where
city in (select city from table-2);
```

b) EXISTS
→ returns either true or false value

```
select * from customers where
exists (select city from table2
        where table2.id = customer.id)
```

when use

IN → Big outer query & small inner query.
EXISTS → small outer query & Big inner query.

Date:
Page

IN → Compare one value to several value.
EXISTS → tells you whether a query returned any results.

(iv) ORDER BY Vs GROUP BY

ORDER BY	GROUP BY
→ sorting	→ used with aggregate functions.
→ ASC or DESC	
→ select * from customers order by ID;	→ select IsActive, Count(*) from customers group by IsActive

* 'GROUP BY' follows 'WHERE' Clause in 'select' statements.

* 'WHERE' can't be used after 'GROUP BY'.

* 'HAVINGS' is used after 'GROUP BY'.

Date:
Page:

(V) JOIN Vs SUBQUERY

→ both are used to combine data from different tables into a single result.

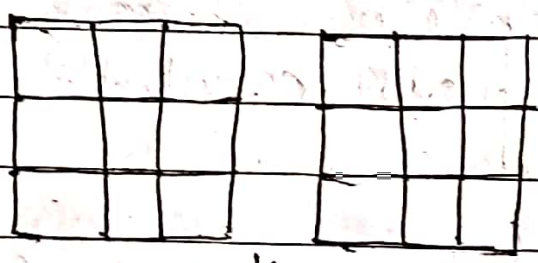
Subquery	Join
→ select phone, cust_name from customers where cust_id IN (select cust_id from orders);	→ select phone, cust_name, order_id from customers C join orders O on C.cust_id = O.cust_id;
→ Can select only from first table.	→ Can select from either of the table
→ Slower	→ faster

Date: _____
Page: _____

(VI) JOIN VS UNION

UNION

→ Combine rows
→ not necessary to have a same column name.



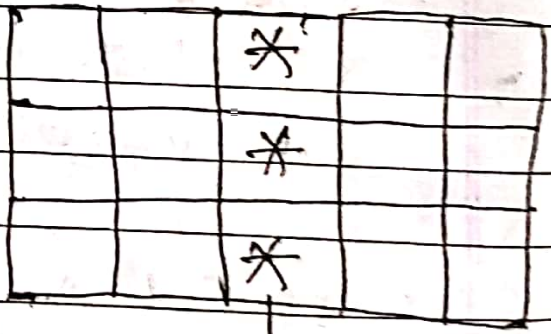
↓ union

→ Select city from table1
UNION
Select city from table2

JOIN

→ merge columns
→ Combine rows from 2 or more tables based on a related common column between them

↓ Join



↓
Common

→ Select a.city, b.name from table1 a JOIN table2 b ON a.Id = b.CustId;