27-06-2025

→ create a user
→ Download the sql file
    → 25, add owner test
    → reopen with window xxxx
    → select window xxxx & save with utf-8
→ login to superuser (postgres)
→ /i 'path of .sql file'
→ In no error :—
    → /q
    → login to superuser with postgres db
    → step 7 : Grant All .... (optional)
    → /q

    → login to superuser with chinook as data base.
    → Grant all three permission P1, P2, P3
    → /q
    → login to test with chinook as db.
        → it not working
            → login to postgres with chinook as db.

→ If error in /i '.sql'

→ login to superuser with postgres as db.
→ delete older .sql file.
→ download .sql file again.
→ open vi, in line 25 : add owner test.
→ follow reload & save instruction.
→ login to superuser
→ drop database chinook with (force);
→ /i '.sql'
→ no error
→ /q
→ login to superuser with postgres db
→ step 7 : Grant All .... (optional)
→ /q

→ login to superuser with chinook as data base.
→ Grant all three permission P1, P2, P3
→ /q
→ login to test with chinook as db.
→ it not working
→ login to postgres with chinook as db.

29-06-2025

Agenda - Database -II

→ Postgres ← psql

→ \i
→ \d
→ \conninfo

→ SQL

SQL

SQL commands are the fundamental building blocks for communicating with (DBMS)

Commands

- TCL (Transaction control language)
- DCL (Data control language)
- DML — Data Manipulation language
- DDL — Data Definition Language
- DQL — Data Query language

DDL – used to define, alter, delete database structure.

- → Create
- → ALTER
- → Rename
- → Drop
- → Truncate

DQL → get data out of database.

- → Select

DML → control access to data & to database. Manipulate.

- → Insert
- → lock
- → update
- → Delete

→ DCL → control access to data in the database by granting or revoking permission.

- → Grant
- → Revoke

→ TCL –

- → Begin Transaction
- → commit
- → Rollback
- → save point

DBA ✗ SQL ✗ DS ✓

SELECT

Retrieve data from one or more tables.

→ SELECT column_name FROM table_name WHERE condition

ORDER BY column_name;

→ SELECT column_name FROM table_name;

→ SELECT * From table_name;

↓

all data,
all column

SELECT + WHERE

where - filter records by condition.

SELECT * FROM table_name WHERE condition;

```
SELECT * FROM album WHERE artist_id=2;
```

SELECT + WHERE + LIKE

LIKE → % → match any sequence

_ → match a single character

ABC
ABCD
ABCDE
AB

} AB

get all the rows where album starts with 'L'

```
SELECT * FROM album WHERE title LIKE 'L%';
```

M → start M%.

end → %M

start → M ← end

↓

%M% → any or none (before & after)

DISTINCT

select only unique values

```
SELECT DISTINCT column_name FROM table_name;
```

```
SELECT DISTINCT artist_id FROM album;
```

COUNT

counts rows

```
SELECT COUNT(*) FROM table_name;
```

```
SELECT COUNT(DISTINCT artist_id) FROM album;
SELECT COUNT(*) FROM album;
```

ORDER BY (SORT)

sort results.

descending ↑

```
SELECT * FROM table_name ORDER BY column_name ASC|DESC;
```

↓ Ascending

→ sort the artist_id in ascending order

```
SELECT DISTINCT artist_id FROM album ORDER BY artist_id ASC;
```

↓ get — ↓ data from

get → all → album → DISTINCT → artist_id → order by artist_id;

SELECT a, b from table_name Order by C; ✓

SELECT DISTINCT C From table_name ORDER BY A;

✗

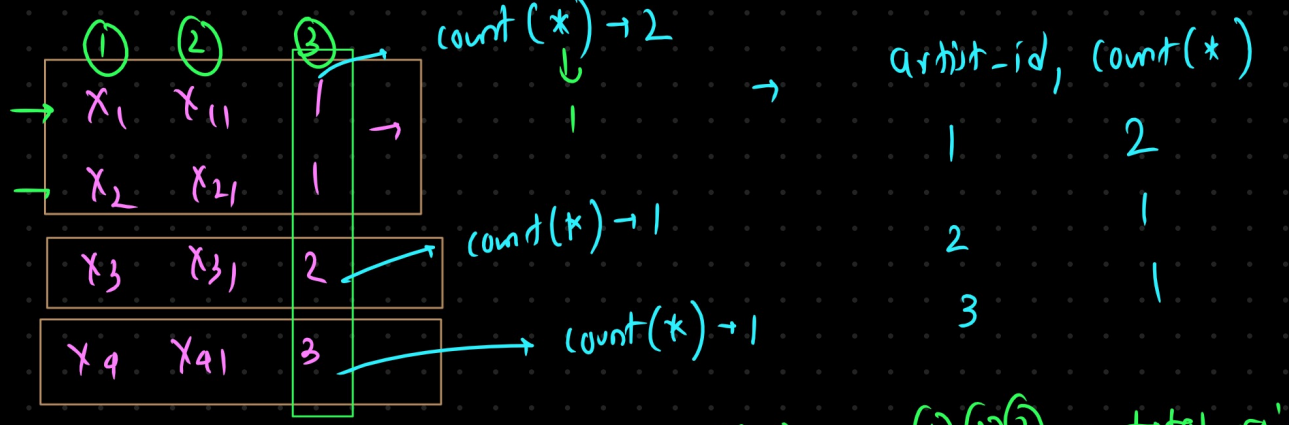
GROUP BY

Aggregate result by column.

Select column_name, aggregate_function() from table_name

group by column_name;

Number of albums per artist.



count(*) ① ② ③ → total row

count(column_name) → total row in that column

```
SELECT COUNT(*), artist_id
FROM album
GROUP BY artist_id
ORDER BY artist_id DESC;
```

AS (Alias)

Rename column or table

Select column_name AS alias from table as alias;

import numpy as np

HAVING

Filter after group by.

WHERE - filter rows before grouping.

HAVING - filters groups after group by.

```
Select artist_id, count(*) From album Group BY artist_id;
```

```
Select artist_id, count(*) From album where artist_id < 100
                                  ↓                ↓
                                 all     Group BY artist_id;
```

```
Select artist_id, count(*) From album where artist_id < 100
                                  ↓                ↓
                                 all     Group BY artist_id;
                                       HAVING artist_id < 10;
```

Count albums per artist and filter artists with > 5 albums

↓

artist_id

```
SELECT artist_id, count(album_id)
FROM album
GROUP BY artist_id
HAVING COUNT(album_id) > 5;
```

```
SELECT artist_id, COUNT(album_id)
FROM album
WHERE artist_id < 100
GROUP BY artist_id
HAVING COUNT(album_id) > 5;
```