Dt:- 21/6/22
10:00Am

SQL

* We Work on "MySQL.
* Standard Data Base Language.
* Detailed
* DEEP

↓

Structured Query Language

* What is the difference between python & SQL ?

Programming Language	Query Language.
• Procedual Language we have to write The total procedure	• Declarative Language. Write Only what You want.
• Line to Line Execution of Program	• No need of Line to Line Execution
• Programming is The main in python, c c++, Java	• No Programming required for Execution.

* Compiler Design :

→ it has "parser" compiler

* parser = break in to parts (Keywords)

Ex:- Select Age from students where age > 22

→ where, This is Query
it uses parser to break into parts

Select	Age	from	Students	where	age > 22

→ Database

→

Students

Name	Age	Educatio	Phone.
	23		
	22		
	25		
	22		
	20		
	18		
	28		

: Here it only consider "Age" column. from Students Data base where age is above / Greater Than 22 :. it ignores other column and less than 22 rows.

* **Structured Data** :-

→ it has proper format (or) structure

→ Data is organized each column

→ it is in "tabular form" .. Excel (or) csv files

→ it stores "Structured Data base"

* **Unstructured Data** :-

→ it has no proper format (or) structure

Ex:- twitter (tweets)

- "hi ☺"
- <3 You much.....
- ☹☹☹

→ it doesn't have any organized way to write in format structure.

→ it stores in "Unstructured database."

"Liver Patient"

Example :-

Age	Gender	TB	Alkphos

* Programming Language :

- Pandas :- df [df ["Age"] < 30]
- Python :- for i in liver Patient :
  If Age < 30 :
  Print (i)

* Query Language :

- SQL :- Select Age from Liver Patient where Age < 30
  ⇓
  * SELECT age FROM Liverpatient WHERE age < 30 ;

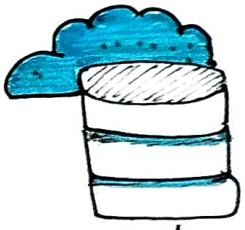
Here we Extract The data
From "Liver patient"
where "Age is Less Than 30".

- (*) ⇒ all columns in SQL
- Select ⇒ selecting The records (which ever You want)
- From ⇒ which Table
- Where ⇒ Condition
- Liver patient ⇒ table name
- Age < 30 ⇒ Writing The Condition (clearly)

it Apply
"Parser + Compiler"
↓ break into parts
↓ apply "code in Backend

* What is Database ?

- A repository (storing) of Data.
- In Database we can add, delete or modify The data by Querying
- Different kinds of data bases store data in different forms
  - EX:- Tabular – structured data base
  - Photos, videos, texts – unstructured data base
- Data stored in tabular form is a "relational data base".



Data

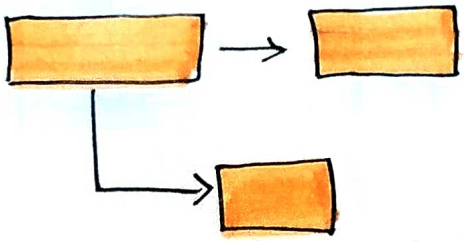
* Benfits

- Ease of use
- Scalability
- Disastery Recover

* what are different of data base ?

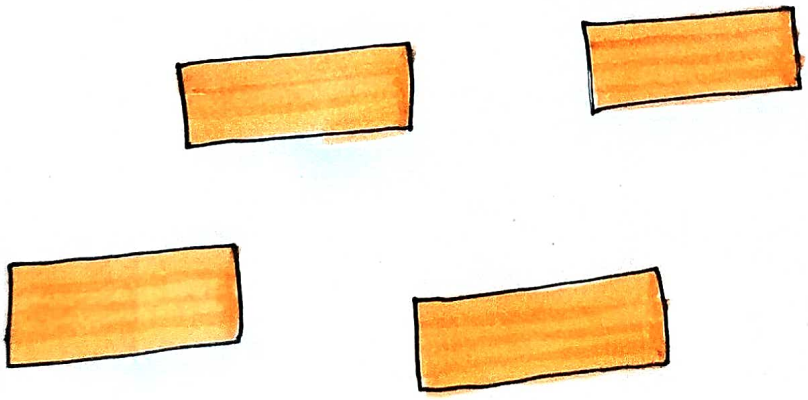
- relational data base
- Non - relational data base

* Relational Database :- "MYSQL"



- Relation (connections) between Multiple tables (with in Database) is called as "Relational database

* Non Relational Database :- "No SQL"



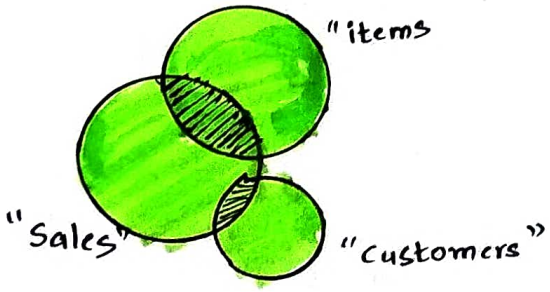
- Doesn't have any Connections between Tables or data.
- it has more complex systems
- Mathematics & Programming.

* We work on "Relational Database".

- Main goal :- Organize huge amounts of Data that can be quickly retrived
  ↓
  Extracted

- solution :- "relational algebra"
  ↓
  fitting data (making) into different Tables and having The connections between Them.

Ex: 3 Tables



* Relational schema ?

"Having the relation between The Tables is called relation schema".

all The connection between Tables

(or)

An Existing idea of how the data base must be organized "relational Schema".

Example :- "movies" Data

Data :-

movies

	movie Id	movie name	Year	actor ID	actor name
0	1	POC	2008	101	Jack Sparrow
1	1	POC	2008	102	Peter
2	1	POC	2008	103	angelino
3	1	POC	2008	104	Sam remi
4	1	POC	2008	105	Amber heard
	⋮	⋮	⋮	⋮	⋮
18	1	POC	2008	119	Robert
19	1	POC	2008	120	Mary

⇒ 20 X 5 = 100 val

⇒ 20 X 3 = 60 values (repeated)

⇒ 20 X 2 = 40 values (Unique)

20 X 3 = 60 Values

unique values

movie

movie Id	Year	movie name
1	2008	POC

1 X 3 = 3 columns

actor

actor Id	actor name
101	Jack Spar
102	Peter
⋮	⋮
119	Robert
120	Mary

20 X 2 = 40 rows

casting

movie Id	actor Id
1	101
1	102
1	103
⋮	⋮
1	119
1	120

"Making connection" between the Tables "relation schema"

" Presentation of Connections (or) relational schema is known as "ER Diagram"

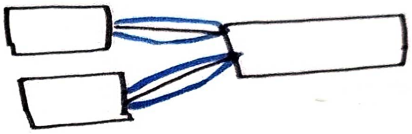
Entity Relation

20 X 2 = 40 rows

Relational Schema

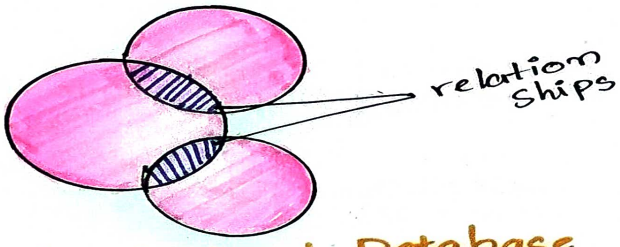
↓ Presented by

E.R Diagram



Entity Relation Diagram

* RDBMS :-



(Relational Database Management System)

" whatever the management, that going to do on t

" Relational Database " is called as " RDMS "

* Relational Database Management System

* Normal table

Customers

ID	first name	Last_name	Email Adress	no. of complaints

* In SQL :

Table name : Customers

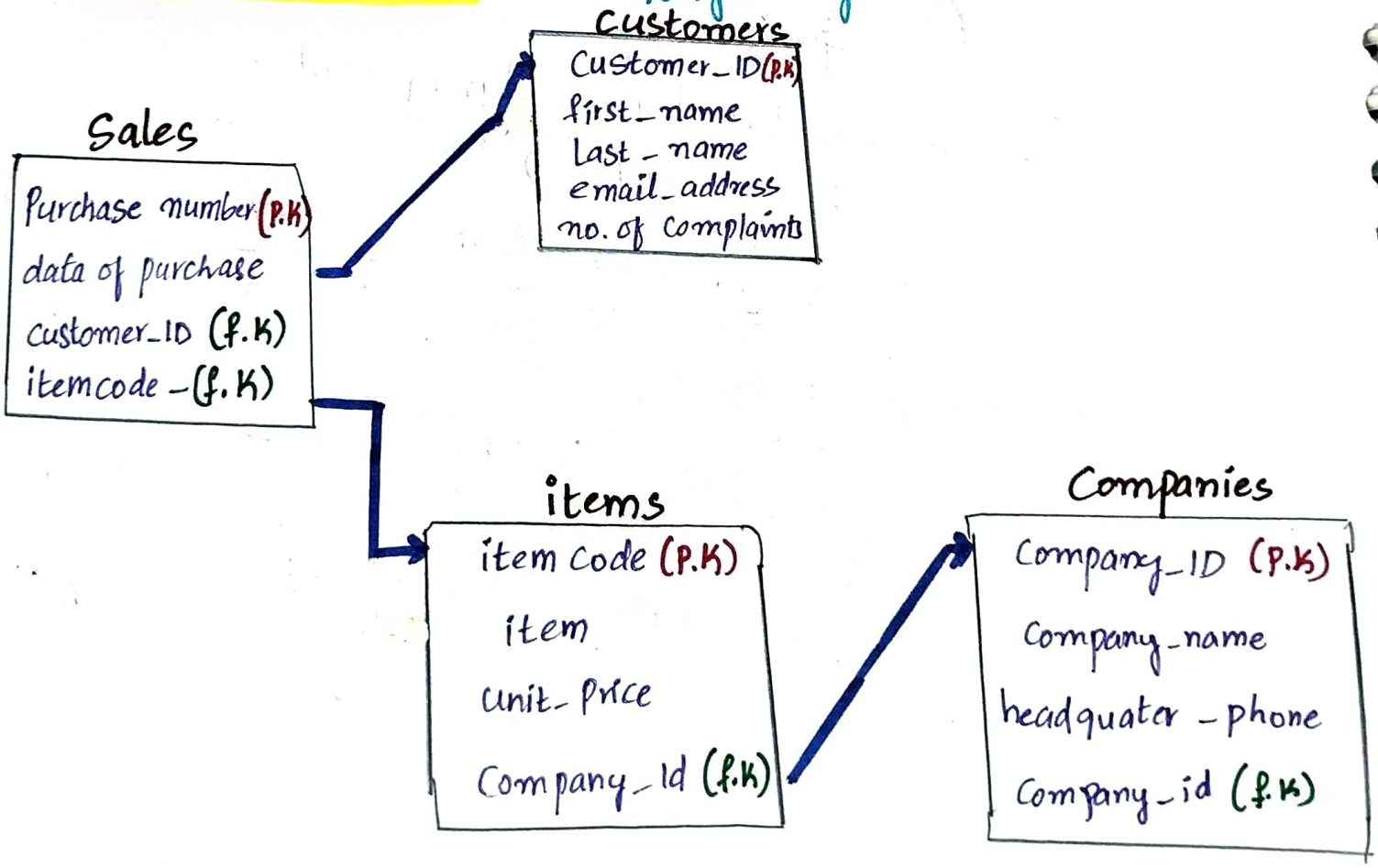
column names

Customer ID
first_name
Last_name
Email_Adress
no. of Complaints

* Database Schema

(P.K) Primary Key (unique Values)

forign Key (f.K)



Example :

movie

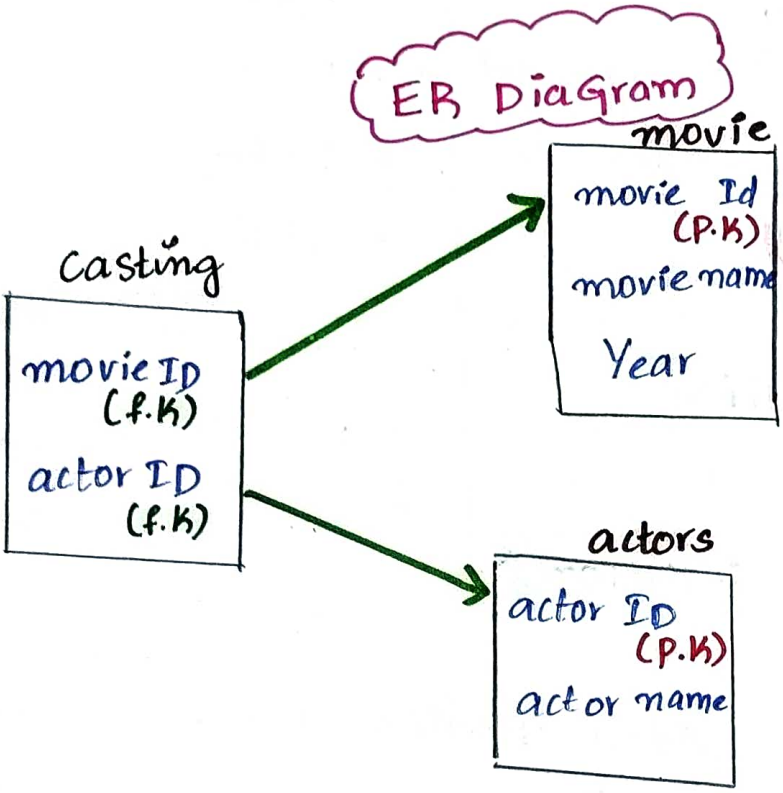
movie Id	movie name	Year

actors

actor ID	actor name

Casting

movie Id	actor Id



• Data Base roles

1. Data base Developers ⇒ (create D.B , Create table , relation Sch
insert the values)
2. Data base Adminstration ⇒ (giving Permission)
cancel Permission
3. Data Engineer ⇒ collect Data from various data sources
and stores in Database.
→ "SQL" Developer

* What is Database Manipulation ?

- Allows You to use your dataset to Extract business insights.
  - Performance
  - Efficiency

* What are SQL Syntax ?

- DDL
- DML
- DCL
- TCL

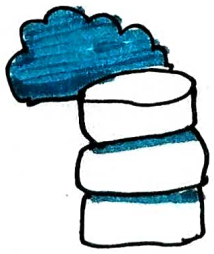
These are The Few Syntax in the SQL.

* What is Database Management ?

- Database Design + Creation + Manipulation

* What is Database Administration ?

- Maintance of Database



* What is The difference between Database and Excel (spread sheets) ?

- Database Vs Excel (spread sheet)

  - Both stored in "Structured format" and "tabular format"
  - In Excel also we can have relationships between Tables :- spread sheet 1, spread sheet 2

Common in Both

relational Data bases	Excel (spread Sheets)

- can contain a large amounts of tabular data
- can use Existing data to make Calculations
- used by many users
- data storage

Differences between

relational data base	Excel spreadsheet

- Pre-set The type of data contained in certain Field

  * data types Fix aye Vuntaye Mundey

Ex:

Name	Age	Educa	Exp
	22 20-1-20		

Particular data type

We can't add anything other than in the column. because it is pre-defined in creating the data base.

↳ Fixed

Ex:- Age(INT)

- we can add any type of data, in Column.

  Endulo data types Em Vundavu. Edina save cheyachu

Ex:-

Name	Age	Edu	Exp
22	30	BTech	2
Sam	20-1-20	BTech	38-1-2

Here, we can add anything that we need by ignore data type of particular column

- all calculations and operations are done after data retrieval (Extracted)

You can do calculation in ("views")

record of data ≠ calculations

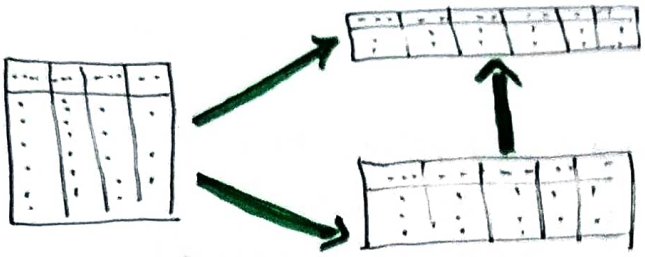
* Extract chesinaka, calculations cheyachu

- different cells can contain calculations (functions and formulas)

* calculations direct cheyachu Excel lo.

- In Database we can connect anything with numerous amount of Tables

* Chala Tables create cheyachu.



- We can create more than 1 million rows ✓.

* Data consistency :

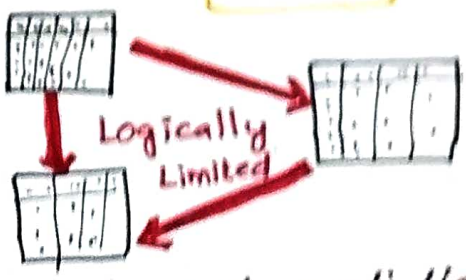
- Provide a Stable Structure, Controlling access Permissions and User restrictions

* EKKada Information change cheyali antey Permission Kavali.
* change chesina vani information kuda save aye thadi.

- Data Consistency
  Data Integrity
  But No duplicate information
  Each Table has unique information

* Duplicates Emi Vundavu, Prati Table unique ga vuntadhi

- Connection between spread Sheet 1 to another is Limited



* Tables create anedi limited.

- 1 million rows maximum

Google docs

Problem :

- finding out who changed or deleted information incorrectly

* Evaru change chesaro, Evadu delete chesado, Evaniki Theliyadu.

- We allow number of duplicates

* We have many duplicates. chala vuntaye.

Advantages of Database:

* Storing and keeping track of Data
* retrieval of Data
* updating of Data
* Efficiency
* Data consistency
* Data integrity
* Speed
* Security

Advantages of Spread sheets

* Extensive analysis

21/06/22

6:00 pm

Dt: 25/6/22
11:00Am

Data types in SQL

* We must always specify the type of data that will be inserted in each column of the Table.

Different data types represent different types of Information that can be contained in a Specific Column

Ex :-

"String"

Surname of a person	Length	Size
"James"	5 Symbols	5 bytes

1 Symbol = 1 byte

* String Data types :-

1. Character [CHAR] → Fixed data type → Max. size (byte 255
   ↳ 50% Faster

2. Variable character [VAR CHAR]
   ↓ not Fixed data type
   lot more responsive to the data value inserted
   Maximum size (bytes) = 65,535

3. enumerate enum

Example:-

String Data types

→ For Text Data

String Data type		Storage	Example			Max. Size (bytes)
character	CHAR	Fixed	CHAR (5)	Length (Symbols)	size (bytes)	255
			"James"	5	5	
			"Bob"	3	5	
Variable Character	VARCHAR	Variable	VARCHAR(5)			65,535
			"James"	5	5	
			"Bob"	3	3	

* Where we use "CHAR" and "VARCHAR" ?

Company id	Company
1	COA
2	COB
3	COC
4	COD

→ Here we use "CHAR(3)" Because, company name is "Fixed with in The 3 symbols.

Here computational Time is Taken less, where compare to "VARCHAR"

Password: [ ]

We use (VARCHAR 16)

Same as "Google pass rd we can add password of "8 to 16" varchar

Because, Here VARCHAR can allow upto 16 elements (or) Symbols

* String data type :-

3. ENUM
("enumerate)

EX :- ENUM (" B.Tech", "M.Tech", "P.HD")

" Mysql " will show an error if you attempt to insert any value different from "B.Tech", "M.Tech", "P.hD".

Graduation

- B.Tech
- M. Tech
- P. H.D

* Integers

↳ For Numeric Data.

- integer
- Fixed-point
- Floating - point

1. integers [INT]

- Whole numbers with no Decimal point
- Eg :- 5 , 10 , -20 , 1,000

• Numeric Data type

Integer	Fixed Point	Floating Point
• integer	• Decimal • Numeric	• FLOAT • DOUBLE

25/06/22

2:00 pm

Dt: 27/6/22
10:00Am

* **Signed** int (Integers) → Default
  - it Contains both positive and Negative Signs
    (+, -)

* **Unsigned** int (integers) → We have mention clearly in query clearly
  - integers allowed only positive signs
    (+)

• INTEGERS

Numeric Data type	Size (bytes)	Min. value (signed / unsigned)	Max. value (signed / unsigned)
tiny INT	1	-128	+127
		0	256
Small Int	2	-32,768	+32,767
		0	65,535
Medium Int	3	-8,388,608	+8,388,607
		0	16,777,215
INT	4	-2,147,483,648	+2,147,483,647
		0	4,294,967,295
BIG INT	8	-9,223,372,036,854,775,808	+9,223,372,036,854,775,807
		0	18,446,744,073,709,551,615

Example :-

Signed int :- Both positive & Negative signed are allowed.

Name [varchar 20]	Age (tinyint)
Ebbu 20	22 1
Nezer	24
David	28
Peter paul	25
VARUN	23
PRUTHVIK	- 18
SATHVIK	22
nithin	30
Rohith	22

Only 4 bytes will be consider in 20 bytes.

1 → Byte.
* In tiny int "1" cell = 1 byte

If one of them are negative. it gives -128 to +127 numbers only.

if there is no Negative Sign The total column Than 0 to 256

* Why not just use BIGINT all The Time ?

So, BIGINT can Increase more "memory", and Computational time. because, If we want store a da that is up to 256 in numbers. we use only "tiny tint", whether it may 'age' (or) phone no. (anything that's what want to store. it taken on 1 byte and Saves the memory and Computational time.

* Precision :-

total no. of digits (including before To Decimal & After To Decimal)

*

Scale :-

After The digits is called as scale

Example:-

Number	Precision	Scale	Decimal
10.523	5	3	[5,3]
36.875	5	3	[5,3]
2.8308	5	4	[5,4]

Decimals

* **Fixed Point** → represents "Exact Values"

When only one digit is specified with in the paranthesis, it will be treated as The precision of data type.

Ex:- 1234567

Decimal (7) , Decimal (7,0)

* Floating Point.

↳ it is used for "approximate values" only

↳ aims to balance between 'range' and precisio

(⇒ "floating)

Example:

Decimal (5,3) 10.5236789	10.524 (In Decimal, it will round it to be total. ignoring more values)
Float (5,3) 10.5236789	10.5236789 (In float, it saves as .it is it will not round it up to whole no.)

* The Main Difference between The "Fixed" and the "Floatr point type is in the way The Value is represented in The Memory of Computer.

• Fixed & Floating - Point Data types

Floating - point data type	Size (bytes)	Precision	Maximum no. of digi
FLOAT	4	Single	23
DOUBLE	8	double	53

Other Useful Datatypes

- **DATE + 🕒 = DATE TIME**

* Used to represent a date in format

YYYY-MM-DD
Year
2022-06-27

As per U.S Date format

* We could save a time :

HH : MM : SS [. Fraction]
23 : 59 : 59 . 999999

24 hr. format

Example :-

25th July 2018 . 9:30 am

"2018-07-25 9:30:00"

2:48 pm : 14 : 48:00

- **DATETIME**

represents The date shown on The calender and the time shown on clock

- **TIME STAMP**

Used for a well-defined, Exact point in time.

• TIME STAMP

↳ representing a moment in time as a number allows you to easily obtain the difference between two TIMESTAMP values

- it counts "total number of seconds"

Example :

Start time : '2022-07-03 10:30:00' UTC

End Time : '2022-07-03 12:00:00' UTC

Here difference is 1½ hr
90 mins = 5,400
1 mm = 60sec

Time STAMP 5,400 (Seconds)

Total Datatypes In SQL

String, date, time datatypes	Numeric Data types
• CHAR	• INTEGER
• VARCHAR	• DECIMAL
• DATE	• NUMERIC
• DATETIME	• FLOAT
• TIME STAMP	• DOUBLE
"Data must be written with in quotes"	Only Numeric Values are written without quotes.

SQL Syntax

- **DDL** = (Data Defination Language)
  - * Creation of Data
  - • (Table create cheyadam)
- **DML** = (Data Manipulation language)
  - * Manipulation of Data
  - • (oka column ni matramey choose chesukodam & Data ni maruvadam)
- **DCL** = (Data Control language)
  - * assignment & removal of permissions to use This Data
  - • (Permission Evadam & cut cheyadam)
- **TCL** = (transaction Control language)
  - * Saving and restoring changes to a database.
  - • (manam 'data" change chesthey andhariki kuda change avuthundhi)

* DDL [Data defination Language]

- syntax
- a set of statements that allows the user to define (or) modify data structures and objects, such as Tables.

- **SQL Commands**

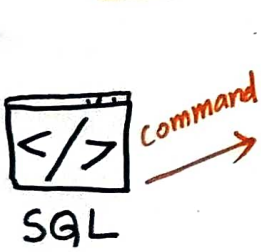
* Data Defination Language [DDL]

1. CREATE
2. ALTER
3. DROP
4. RENAME
5. TRUNCATE

* **CREATE** statement :

Used for creating entire databases and database objects as tables.

- CREATE DATABASE Ebby ;



- CREATE TABLE Object_name (column_name data_type);

Ex:- CREATE TABLE Sales (Purchase_number INT);

Here we can add multiple column when we create table,
Ex:- Date of purchase DAT

Sales

Purchase_number

∴ The Table name can Coincide with name assigned to database.

* The ALTER statement :

↳ Altering / modifying already using (or) Existing objects.

1. ADD
2. REMOVE
3. RENAME

SQL

- ALTER TABLE sales
- ADD COLUMN date_of Purchase DATE ;

Sales

Purchase_number	date_of_Purchase

* The DROP statement :

↳ deleting entire data table.

SQL

- DROP object_type object_name ;
- DROP TABLE Customers ;

Customers

customer_Id	first_name

* it drops total data.

* The RENAME statement :

↳ changing names of Table, col, rows

- RENAME object_type Object_name TO new_object_name ;
- RENAME TABLE customers TO Customer_data ;

SQL

customers

Customer_Id	first_name

RENAMED ⇒

Customer_data

Customer_Id	first_name

* The TRUNCATE statement :

↳ Instead of deleting or droping The Table. and Continue we can remove data in The table. to have Table as an object in The database.

SQL

- TRUNCATE object_type object_name ;
- TRUNCATE TABLE Customers ;

customers

Customers_Id	first_name
.....	.....
.....	.....
.....	.....
.....	.....

truncate is helpful in using the same columns and for New Data.

* What is difference between "DROP", "DELETE", "TRUNCATE"

- DROP : Drop's The Entire Data/table.
- DELETE : Deleting The no. of rows in data, selected data
- TRUNCATE : Truncate helps in reusing The Existing table after deleting old data. by replacing New Data.

* Data Manipulation Language (DML)

↳ its statements allows us to Manipulate The data in the tables of a Database

1. SELECT
2. INSERT
3. UPDATE
4. DELETE

* SELECT

↳ used to retrieve data from database objects Like Tables.

* Why are we going to need just a piece of the table ?

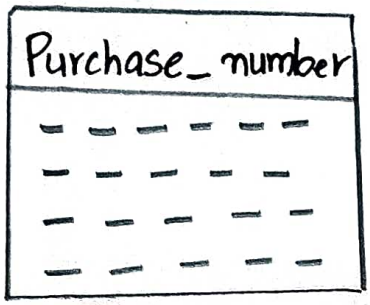
- Imagine a table within 2 millions rows of Data.
- it can be helpful, if you could Extract only a Portion of the table that Satisfies given Créteria.
- You Should Know how to use SELECT perfectly well

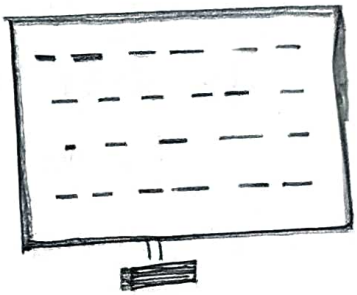
all The columns

```
SELECT * FROM Sales ;
```

SQL

Sales





* The INSERT statement :

↳ used to insert data into tables

- INSERT INTO .... VALUES... :

when ever We have Word "INSERT" we have to choose another word "INTO" with it.

SQL

- INSERT INTO sales (Purchase_number , date_of_Purchase)

  VALUES (1, "2017-10-11");

Sales

Purchase_number	date_of_Purchase

⇒

Sales

Purchase_number	date_of_purchase
1	2017-10-11

27/06/22 6:30Pm

Dt: 29/06/22
11:00Am

* The UPDATE statement

↳ allows you to renew existing data of your Tables (or) replace

↳ along the UPDATE, we have to use "SET" syntax mandatory.

SQL

- UPDATE sales
- SET date_of_purchase = "2017-12-12"
- where Purchase_number = 1

Sales

Purchase_number	date_of_Purchase
1	2017-10-11
2	2017-10-27

⇒

Sales

Purchase_number	date_of_Purchase
1	2017-12-12
2	2017-10-27

* The **DELETE** statement

↳ functions similarly to the "truncate" but Delete only user defined (or) particular records only.

SQL

- DELETE FROM sales
- WHERE

  Purchase_number = 1;

Sales

Purchase_number	date_of_purchase
1	2017-10-11
2	2017-10-27
3	2017-10-28

DELETED ⇒

Sales

Purchase_number	date_of_purchase
2	2017-10-27
3	2017-10-28

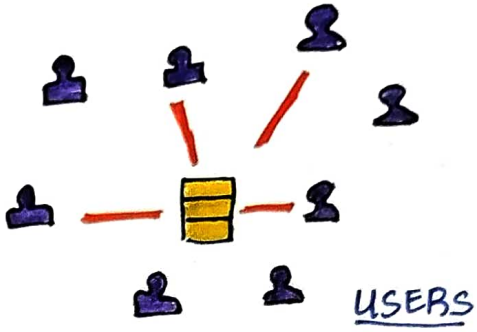
DML Commands used with other words.

- SELECT .... FROM...
- INSERT INTO .... VALUES...
- UPDATE ... SET ... WHERE...
- DELETE ... FROM ... WHERE....

* Data Control Language [DCL]

↳ allows us to manage the rights users have in Database.

1. GRANT
2. REVOKE



* The GRANT statement

↳ Gives (or) Grants Certain Permission To Users.

↳ One Can Grant a Specific type of Permission, like

"Complete" (or) "partial"

(Complete excess) Edit and modify data

(Only view (or) no modifying data)

↳ These rights will be assigned to a Person who has a Username registered at The Local Server ("Local host : IP 127.0

IP Address

Mysql

User root

Password *****

↳ big Companies and Co-operations don't use this type of Server, and their databases lay on External, more Powerful Servers.

SQL

- GRANT type_of_permission ON database_name.table_name TO "Username" @ "Localhost"

* the REVOKE clause
  - ↳ used to revoke (remove) permissions and privileges of database Users
  - ↳ it is Exact opposite of GRANT

  * Cancel The ↑

SQL

- REVOKE type_of_permission ON database_name.table_name FROM "Username" @ "Local host"

* Transaction Control Language [TCL]
  - ↳ not Every change you make to a database is Saved automatically

1. COMMIT
2. ROLLBACK

* The COMMIT statement

↳ related To INSERT, DELETE, UPDATE

↳ will Save The Changes you've made

↳ will let other users have access to the modified version of database.

Ex:-

- change The last name of "4th" customer from "Jack" to "Sam"

Customers

customer_Id	first_name	last_name
1	Alenner	Venmu
2	aymwos	maywosaita
3	amicus	Sucima
4	Prudhvi	Jack

⇒

Customers

Customer_Id	first_name	last_name
1	Alenner	Venmu
2	aymwos	maywosaita
3	amicus	Sucima
4	Prudhvi	John

Here To change data, we use "Update" command in SQL

SQL

```
• UPDATE customers
• SET last_name = "John"
• WHERE Customer_Id = 4;
```

→ it only changes "data" in our systems

• COMMIT ;

→ it will update for Everyone. whose is using particular "Database"

* The **ROLLBACK** clause.

↳ The clause that will let you make a step backs

Ex:- Ctrl + Z

↳ allow you to Undo any changes you have made but don't to be Saved Permanently

↳ Stores "temperolly"

SQL

```
• UPDATE customers
• SET Last_name = "John"
• WHERE Customer_Id = 4
• COMMIT ;
• ROLLBACK ;
```

→ while we writting query "rollback" we can undo the command. Lekapothe. "Commit" anedi direct save ayethadi permanent ga ♡

Customers

customer_Id	first_name	Last_name
1	Alenner	Vennu
2	aymurs	maywosaita
3	amicus	Sulima
4	Prudhvi	John

⇒

Customers

Customer_Id	first_name	Last_name
1	Alenner	Vennu
2	aymos	maywosaita
3	amicus	Suicma
4	Prudhvi	Jack

By writting "Roll back" we can undo The data. that is changed ago.

29/06/22

3:45 pm.

Dt:- 30/6/22
11:00 Am

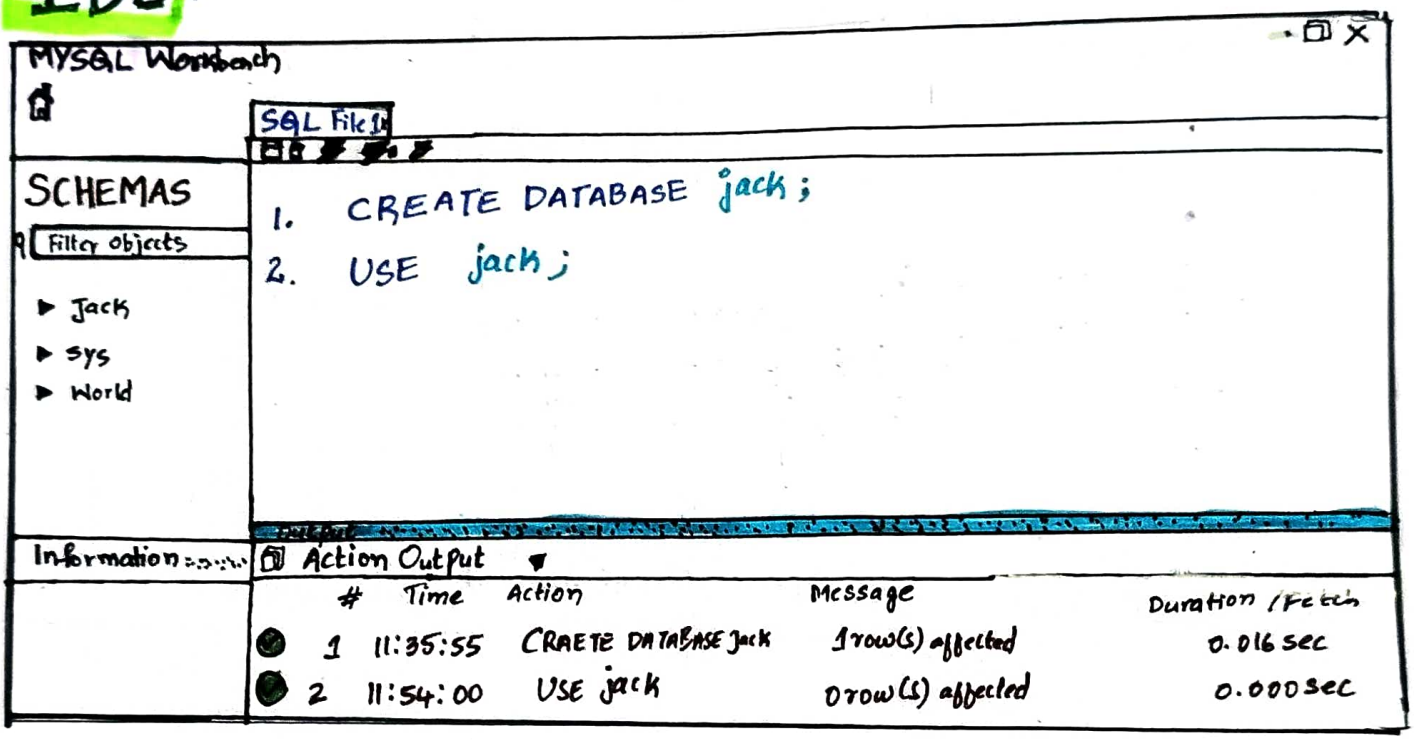
MySQL

* Where we work and Execute queries on SQL ?

In MySQL we have

1. MySQL 8.0 Command Line client
2. MySQL Workbench 8.0 CE [IDE]

IDE :



Example :- Real time "Imdb" data

1. CREATE DATABASE Imdb; → first database "create" cheyali
2. USE Imdb; → dhanni use cheyali
3. SOURCE Path;

Imdb.sql file ekkado vundo dhani path rayali.

SQL is not case sensitive, it will accept lowercase, uppercase

* Use lmdb;

* SHOW tables;

But, commands should be in upper case only

Out:

	tables_in_imdb
▶	actors
	directors
	directors_genres
	movie
	movies_directors
	movies_genres
	roles

Action	Message
Show tables	7 row(s) returned

* DESCRIBE directors_genres;

Out:

	Field	Type	Null	Key	Default	Extra
▶	directors_Id	int	NO	PRI	Null	
	genre	varchar(m)	NO	PRI	Null	
	Prob	float	YES		Null	

Primary Key

· Action

DESCRIBE directors_ge

· Message

3 row(s) returned

Ex:- "ER" Diagram

* DESCRIBE movies_directors;

Out:

	Field	Type	Null	Key	Default	Extra
▶	directors_Id	int	NO	PRI	Null	
	movie_Id	int	NO	PRI	Null	

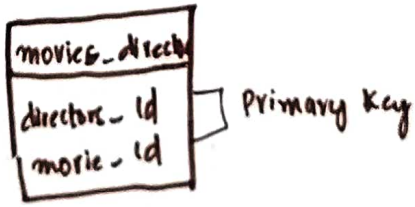
Action

DESCRIBE movies_

Message

2 row(s) returned

Ex:- ER Diagram



30/6/22 1:00 pm

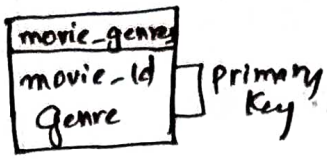
Dt:- 1/7/23
11:00 AM

* DESCRIBE movies_genres ;

Out:

	Field	Type	Null	Key	Default	Extra
▶	movie_id	int	No	PRI	Null	
	genre	varchar(100)	NO	PRI	Null	

Ex: ER Diagram



Action
DESCRIBE movie_genres

Message
2 row(s) returned

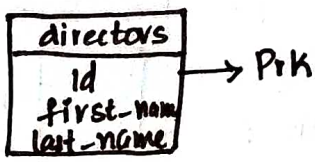
* DESCRIBE directors ;

Out:

	Field	type	Null	Key	Default	Extra
▶	id	int	No	PRI	0	
	first_name	varchar(100)	YES	MUL	Null	
	Last_name	Var char(100)	YES	MUL	Null	

multiple → Normal key

Ex: ER Diagram



Action
DESCRIBE directors

Message
3 row(s) returned

* DESCRIBE movies ;

Out:

	Field	Type	Null	Key	Default	Extra
▶	id	int	No	PRI	0	
	name	varchar(100)	YES	MUL	Null	
	year	int	YES		Null	
	rankscore	float	YES		Null	

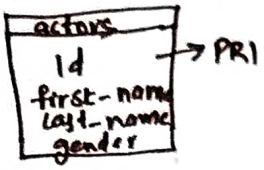
Ex: ER Diagram

* DESCRIBE actors ;

out :

	Field	Type	Null	Key	Default	Extra
▶	id	int	No	PRI	0	
	first_name	varchar (100)	YES	MUL	Null	
	last_name	var char (100)	YES	MUL	Null	
	gender	char (1)	YES		Null	

Ex:- ER Diagram



Action
DESCRIBE actors

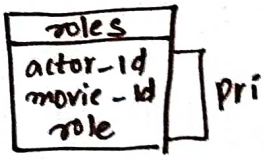
Message
4 row(s) returned

* DESCRIBE roles ;

out :

	Field	Type	Null	Key	Default	Extra
▶	actor_id	int	No	PRI	Null	
	movie_id	int	No	PRI	Null	
	role	varchar(100)	No	PRI	Null	

Ex :- ER Diagram



Action
DESCRIBE roles

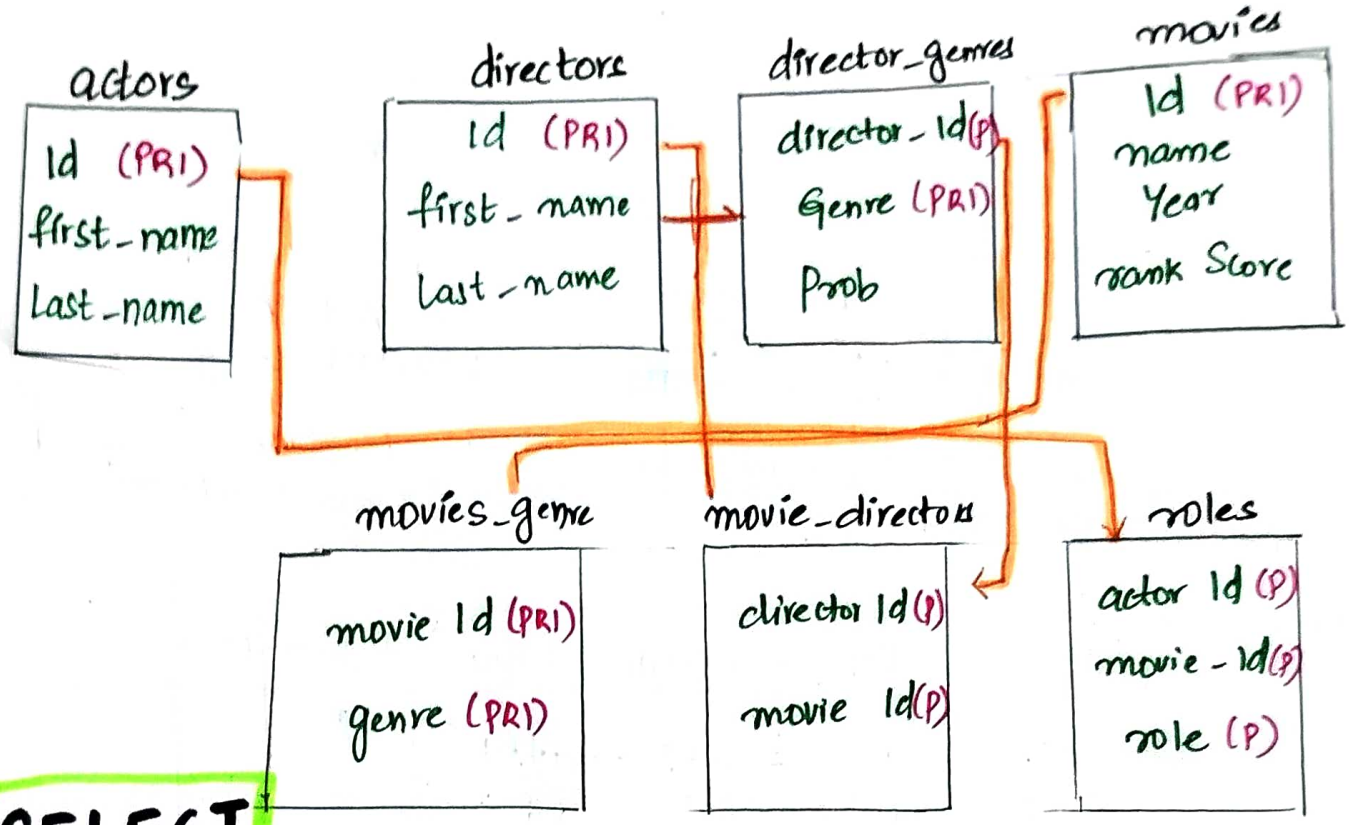
Message
3 row(s) returned

Q:- Can we Draw The ER Digram and Relational Schema for The following "Imdb" Data ?

"Imdb"

1. actors
2. directors
3. directors_genres
4. movies
5. movies_directors
6. movie_genres
7. roles.

ER Digram & Relational Schema For "Imdb" Data.



• SELECT

* DESCRIBE movies ;

* SELECT * FROM movies ;

→ all columns

Out :

	id	name	year	ranksore
▶	0	# 28	2002	Null
	1	# 7 Train: An Immigrant	2000	Null
	2	$	1971	6.4
	3	$ 1000 Reward	1913	Null
	⋮			

Action

SELECT * FROM movies

Message

388 274 row(s) return

* Use imdb ;
* From these it gives the

"SELECTED" in "movies"
total 4 columns
and total rows

for only two column (or) User-defined Columns

* SELECT name, rankscore FROM movies;

Out:

	name	rankscore
▶	#28	Null
	#7 Train: An Immigrant	NULL
	$	6.4
⋮	$1000 Reward	NULL

Action

SELECT name, rankscore FROM movies

Message

388269 row(s) returned

LIMIT

* SELECT name, rankscore FROM movies LIMIT 20;

Out:

	name	rankscore
▶	#28	NULL
	#7Train: An Impr	Null
	⋮	⋮
	$40,000	9.6
	$5,000 Reward	Null

(Action)

Limit ⇒ The first "20" records

Message

20 row(s) returned

OFFSET

* SELECT name, rankscore FROM movies LIMIT 20 OFFSET 40;

Out:

	name	rank score
▶	'60s Pop Rock Reunion	Null
	'60's, The	Null
	⋮	⋮
	"As he hizo" Torremolinos 73	Null

Action:

Delete The First 40 rec and From print 20 records

MESSAGE:

20 row(s) returned

ORDER BY

* SELECT name, rankScore, year FROM movies ORDER BY year LIMIT 10;

starting year @ 1888

out:

	name	rankScore	Year
▶	Roundhay Garden Scene	Null	1888
	Traffic crossing Leeds Bridge	7.3	1888
	⋮	⋮	⋮
	Monkey and another, Boxing	3.2	1891
	Duncan and Another, Blacksmith Shop	3.5	1891

We want year wise order by lowest to max

Sequential order with limit of "10"

only 10 records must be printed

Message
10 row(s) returned.

∴ By default it takes Ascending order.

DESC

* SELECT name, rankScore, year FROM movies ORDER BY year DESC LIMIT 10;

out:

	name	rankScore	Year
▶	Harry potter and Half-Blood Prince	Null	2008
	Tripoli	Null	2007
	⋮	⋮	
	Andrew Henry's Meadow	Null	2006
	American Rain	Null	2006

Desc;
Gives The Descending order of The data

Message
10 row(s) returned

DISTINCT

* SELECT DISTINCT genre FROM movies_genres;

out:

	Genre
▶	Documentary
	Short
	Comedy
	⋮
	Film Noir

Distinct;
"Unique" values from movie_genre "genre" column

Message
21 row(s) returned

WHERE

SELECT name, year, rankscore FROM movies WHERE rankscore > 9 ;

Condition it is going to Select where rankscore is Greater Than 9

Out:

	name	year	rankscore
▶	$ 40,000	1996	9.6
	+1 - 1	1987	9.6
	⋮	⋮	⋮
	Im Savasisi	1984	9.7
	Itima Ilamada, La...	1996	9.5

Message
1069 row(s) returned

We write Condition "where", based on Condition, which records satisfies the Condition, only those will get printed.

SELECT name, year, rankscore FROM movies WHERE rankscore > 9 ORDER BY rankscore DESC ;

Print The, where rankscore gr Than 9. From movies in descending order.

Out:

	name	year	rankscore
▶	Atundi iam- comdamant	1971	9.9
	Blow job	2002	9.9
	⋮	⋮	⋮
	Zink Somoty Berhof	1982	91
	tat critique	1992	9.1

Message
1069 row(s) returned

* SELECT * FROM movies_geners WHERE genre = "Comedy" ;

Out:

	movie_id	genre
▶	2	comedy
	6	comedy
	8	comedy
	⋮	⋮
	378613	comedy
	378614	comedy

Print, where genre = "Comedy" From movie_generes

In ER Diagram we have only 2 columns in movie_genres

Message
56425 row(s) returned

are comedy movies

From 388269 rows

Comparsion Operations :

- = (Equals to)
- <> (or) != (not Equals to)
- >= (Greater Than Equal to)
- <= (Less Than Equal to)

* SELECT * FROM movies-genres WHERE genre <> "Horror";

Out :

	movie_id	Genre
▶	1	Documentary
	1	Short
	2	Comedy
	⋮	⋮
	388613	Drama
	378614	Comedy

it returns, where not equals to horror remaining Everything it is printing

Message

388078 row(s) returned

* SELECT name, year, rank Score FROM movies WHERE rankScore IS NULL;

Space Vundali

Out :

	name	year	rankscore
▶	#28	2002	NULL
	#7 train : An Immigrant	2000	NULL
	⋮	⋮	⋮
	"Istanbul"	1983	NULL
	"Sterreich	1958	NULL

Is Null gives The Only null values in the Certain Column of rankscore

Message

321024 row(s) returned

* SELECT name year rankscore FROM movies WHERE rankscore IS NOT NULL

Out :

	name	Year	rankscore
▶	$	1971	6.4
	$1,000,000 Duck	1971	5
	⋮	⋮	⋮
	nc. sayfa	1999	7
	.19.99	1998	6.3

Is NOT Null gives values That are not null values in the Column of rankScore

Message

67245 row (s) returned

Date:- 5/7/22
11:00Am

LOGICAL OPERATORS

- AND
- OR
- NOT
- BETWEEN
- IN
- EXITS
- LIKE
- SOME

AND

* SELECT name, year, rankscore FROM movies WHERE rankscore >9 AND year > 2000 ;

Out:

	name	year	rankscore
▶	12 (2003/II)	2003	9.8
	14 million Dreams	2003	9.5
	⋮	⋮	⋮
	Ziti hui mie	2001	9.6

AND ⇒ it should meet The both The conditions. where, rankscore should be above (Greater Than 9) AND year should be Greater Than 2000.

Message
250 row(s) returned

NOT

* SELECT name year rankscore FROM movies WHERE NOT year > 2000 LIMIT 10 ;

Out:

	name	year	rankscore
	# 7 Train : An Imagina	2000	Null
	$	1971	6.4
	$ 1000 record	1913	Null
	⋮	⋮	⋮
	$ 1000 a Touchdown	1939	6.7

NOT, it should satisfy That particular Condition. where year should not be greater Than 2000, and Limit up to 10 records

Message
10 row(s) returned

OR

* SELECT name, year, rankScore FROM movies WHERE rankScore > 9 OR year > 2000 ;

Out:

	name	Year	RankScore
▶	# 28	2002	5.4
	$ 100,000 Pyramid	2001	6.8
	$ 300 y ticket	2002	Null
	⋮	⋮	
	"zgnm Leyla"	2002	Null

OR ⇒ Satisfy any one condition from given condition
Here, it taken Year Greater Than 2000, and left out rankScore, where we can see 5.4, but condition greater Than 9.

Message:
46825 row(s) returned

BETWEEN

* SELECT name, year, rankScore FROM movies WHERE year BETWEEN 1999 AND 2000 ;

In PanDas = year >= 1999 and year<=2000

Out

	name	year	RankScore
▶	# 7 Train : An Sin	2000	Null
	$ 30	1999	7.5
	$ pent	2000	4.3
	⋮	⋮	⋮
	"Halloween" unmasked	2000	Null

BETWEEN ⇒ it gives The what ever satify condition.
Here, movies from 1999 to 2000. only between only These movies will display.

Message
22619 rows returned

* For example,
what happens. if we write.

* SELECT name, year, rankscore FROM movies WHERE year BETWEEN 2000 AND 1999 ;

Out: it Gives Empty Cell.
Because low value <= high value

Nduku antey 2000 nundi 1999 ki back condition satify kadu raphaka

IN

* SELECT director_id, genre FROM directors - genres WHERE genre IN ("Comedy", "Horror") LIMIT 10;

Out:

	director-id	genre
▶	8	comedy
	10	comedy
	23	horror
	⋮	⋮
	41	horror

MESSAGE
10 row(s) returned

IN = same as genre = "Comed OR genre = "Horror

Aggregate Functions

- MIN
- MAX
- COUNT
- SUM
- AVG

MIN

* SELECT MIN (year) FROM movies;

- In pandas : movies.df["year"].min()

Out:

	Min(Year)
▶	1888

Message
1 row(s) returned

Min, gives the minimum value for the given condition.
Here, we see MIN(year) it given one output 1888 has min year from movies.

5/7/22
1:20 pm

PE:- 6/7/22
11:00 Am

MAX

* SELECT MAX(year) FROM movies ;

Out:

	MAX (year)
▶	2008

MESSAGE
1 row(s) returned

Max : it is opposite of "MIN" - which gives Maximum value According to condition.
Here, 2008 is the Maximum Year in movies ☺

COUNT

* SELECT COUNT (*) FROM movies ;

Out:

	COUNT(*)
▶	388269

Message:
1 row(s) returned

Count : → Gives no. of records.
Emlee bro. Count chepthadi motam.
Kakapothey oka magic Enti antey. "NULL" value ni ignore chesthadi

* SELECT COUNT (year) FROM movies WHERE year > 2000;

Out:

	COUNT (year)
▶	46006

Message:
1 row(s) returned

2000 year Kanna Edi ayethey Vuntadho anni print chesthadi.
Eppudu 2000 year akka Vunnaye 46006 anamata ♡

GROUP BY

So, To understand Clearly
We Have

Ex:-

2009
2010
2009
2008
2009
2010
2008
2000

2009 - 3 records.
2010 - 2 records
2008 - 2 records
2000 - 1 records

Antey. Same Vunna vatini oka ga chesi, Vati value count esthadi. dhanni GROUP Antaru Mawa.

* SELECT year, COUNT(year) FROM movies GROUP BY year ;

Out :

	year	COUNT(year)
▶	2002	12056
	2000	11643
	1971	4072
	⋮	⋮
	1882	2

Message
120 row(s) returned

it Gives Values counts of for The Given condition. In simple words

* SELECT year, COUNT(year) FROM movies GROUP BY year ORDER BY year ;

Out :

	year	COUNT(year)
▶	1882	2
	1890	3
	1891	6
	⋮	⋮
	2008	1

Message
120 row(s) returned

Here, ORDER By is added with GROUP By. So, it gives order wise from min to MAX.

HAVING

Q:- what is The difference between WHERE and HAVING ?

* WHERE :- it is Applicable "directly to Condition on column."

* HAVING :- it can applicable only after applying "GROUP BY"

without we can't apply HAVING.

After that we have to write condition.

We can write, year-count also

* SELECT year, COUNT (year) count_year FROM movies GROUP BY year HAVING count-year > 10000 ;

Out :

	year	count_year
▶	2002	12056
	2000	11643
	2001	11690
	1999	10976
	2003	11890
	1998	10067

Message
6 row(s) returned.

Here, HAVING applied on condition [count-year Greater Than 10,000] after GROUP By Function. it Gives year, which has Greater count First.

Joins

- SQL joins Statements allow us to access information from "Two" or more Tables at once.
- They also keep our database normalized

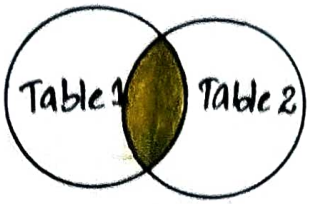
⇒ We have Different types of Joins.

* INNER JOIN
* LEFT JOIN
* RIGHT JOIN
* FULL JOIN

* In Pandas :- We use "Merge" to join The two (or) More Columns.

• INNER JOIN

. RETURNS dataset that have matching Values in both sides (tables)



• INNER JOIN.

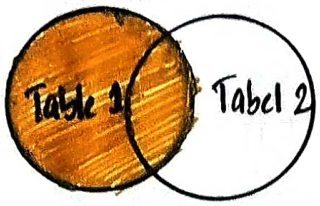
SQL

```
SELECT column_name FROM table 1 INNER JOIN table 2
ON table 1. Column_name = table 2. Column_name ;
```

Condition

• LEFT JOIN

. Returns all records from The left table and matched records from right



• LEFT JOIN

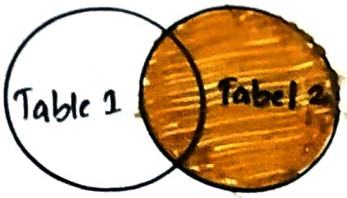
SQL

```
SELECT (*) FROM tabel 1 LEFT JOIN table 2 ON
table 1 . column_name = table 2. column_name ;
```

* → total columns

• RIGHT JOIN

• Returns all records from The right table and matched records from The Left.



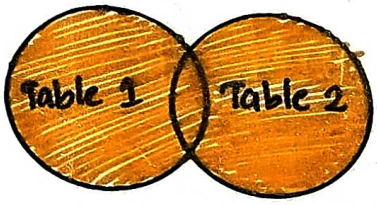
- RIGHT JOIN

SQL

```
SELECT (*) FROM Table 1 RIGHT JOIN Table 2 ON
table 1.* = table 2.*
```

FULL JOIN

Returns all records when There is a match in either the left table or right table



- FULL OUTER JOIN
- CROSS JOIN

SQL

```
SELECT column_name FROM table 1 FULL OUTER JOIN tabl
ON table 1. column-name = table 2. column_name
WHERE condition ;
```

6/7/22
6:00pm

7/7/22
11:00 Am

Q:- So, Why These joins are important ra mawa ?

Dive Deep into joins.

* Asalu "Cartesian product" antey enti

→ Concept of Cartesian product :-

Set that is Constructed from two given Sets and Comprises all pairs of elements such that the first element of the pair is from the first Set and the Second is from the Second Set.

Sollu la Vundhi . . . ?

So, Example :-

Edi Easy ga ardham ayethadhi

Where, we have P, Q ⇒ (P) X (Q)

Cartesian product

P \ Q	a	b	c
1	(1xa)	(1xb)	(1xc)
2	(2xa)	(2xb)	(2xc)

anthey, This is what "Cartesian Product".

Enko Example : Assume.

In SQL, we have two tables

A.

a_1	a_2
1	2
3	4
5	6

B.

b_1
7
8

[A x B] ⇒

a_1	a_2	b_1
1	2	7
1	2	8
3	4	7
3	4	8
5	6	7
5	6	8

"Cartesian product"

So, Joins ki "Cartesian" Product ki Link enti ra mawa :) antey. Ekkada manam oka Table lo vunna data ni. Enko Table Tho cross multiply (or) Cross product chesinam.

So, ala kakunda (AxB) vunna dhaniki I mean original "cartesian product" ki emina condition apply chesthay

Adey Joins

Ex :- $(A \times B)_{A=D}$

⇓

Ekkada (AxB) dhaniki A=D aney condition anedi Cartesian product. rasinam. antey ah Tables lo "A" is equal to "D" vatiney output lo print chesthadhi

* "Cartesian product" followed by any condition is called join.

→ Condition ("cartesian product) = Join

* **Natural join** :

• Natural join operation that creates an implict join clause for you based on the common columns (or) Attributes in two tables being joined.

EX:

P

A	B	C
1	a	b
2	c	d

Q

A	D
1	d
3	e

Ekkada, Evi Rendu same rows (or) Tuples Levu Kabbati. avi join (Natural) lo Enter kavu.

(P) natural join (Q)

A	B	C	d
1	a	b	d

⇐ Out put

* **Left Outer join**

(P) Left outer join (Q)

antey....? (P) Left outer join (Q) ⇒ "Q" ki Left anti "P" so dhani ki condition apply chayali.

Example :-

P

A	B
1	a
2	c

Q

C	D
1	b
3	d

Condition

So, Here $\boxed{\text{P LEFT OUTER JOIN Q}}_{A=C}$

Ekkada, "Q" ki Left "P" kabati, P = Vunna data Motam Print avuthadhi. Ekkada, A=C anedi condition ra mawa. Ekkada, "A" = 1, "C" = 1 same Vunnaye. Kabati theseynimpu. "A" = 2, " = 3 Vundhi. antey not equal so "sollu" (sorry) "Null" adhi.

Output :-

A	B	C	D
1	a	1	b
2	c	NULL	NULL

SQL

```
SELECT(*) FROM P
LEFT OUTER JOIN Q
ON P.A = Q.C ;
```

P.nundi A = Q nundi C

* Right outer join

Same Example.

P

A	B
1	a
2	c

Q

C	D
1	b
3	d

$\boxed{\text{P RIGHT OUTER JOIN Q}}_{A=C}$

So, Ekkada "P" ki right antey "Q" kabati motam nippey A=C aney condition. "C" = 1 Vundi, "A" = 1. So, nippey ad and malli "A" = 2, "C" = 3. antey paxxka poye aduko ani cheppi "NULL" rayali

Output :

A	B	C	D
1	a	1	b
NULL	NULL	3	d

SQL

```
SELECT(*) FROM P
RIGHT OUTER JOIN Q
ON P.A = Q.C ;
```

* **FULL Outer join** SQL SELECT(*) FROM P FULL OUTER JOIN Q ON P.A = Q.C ;

Same Example:

P

A	B
1	a
2	c

Q

C	D
1	b
3	d

P FULL OUTER JOIN Q A=C

Edi Fun Vuntadhi. Ekada, "P" lo Vunna Motam Values Vundali But, A=C dhaggara match Em avuthunaye, First row (tuple) avi nimpali, Second row Same Levu Kabati avi "NULL". Eppudu "Q" Kuda anni Vundali Kadha. anni Vachaye only "3" "d" raledu Rasey.

Output:

A	B	C	D
1	a	1	b
2	c	NULL	NULL
NULL	NULL	3	d

A=C match avvadam Ledu Kabati miginaye "NULL". nduku dhani corresponding Values Levu Kabati.

So, **INNER JOIN**? antey enti. SQL SELECT(*) FROM P INNER JOIN Q ON P.A = Q.C ;

P INNER JOIN Q A=C

Same example.

P

A	B
1	a
2	c

Q

C	D
1	b
3	d

Ekkada, Only Edi ayethey. Condition meet avuthundo. Adhi matramey Print avuthundhi. Output. migitha anni Print avvavu. nduku antey, 2, 3 Same Kadu, Match Kaledu.

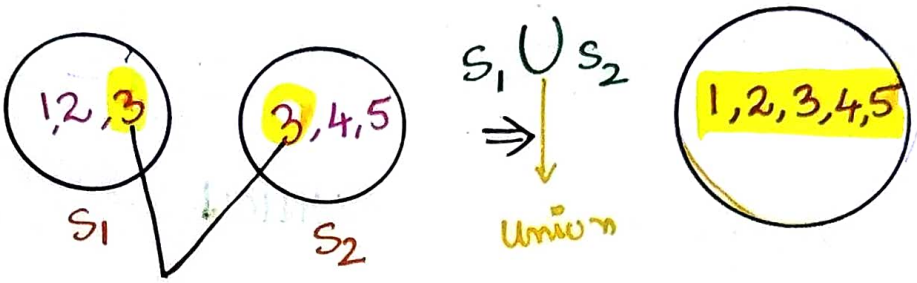
Output:

A	B	C	D
1	a	1	b

Union :- Removes duplicate Records and Combine Them.

Q. What is union ?

A: Combine The result set of two (or) more SELECT Statements.



Common Vunnaye kabati only vatini combine chesthanu.

* "UNION" have Conditions To Apply :

⇒ UNION operator Combines result set of 2 (or) more SELECT statements

⇒ Each SELECT statement Must have.

Ex:- Table 1 [3col] Table 2 [3col]

- Same number of Columns
- Columns must have similar Data types
- Columns must be In Same Order

Example.

CUSTOMERS

Cust_Id	Address	Contact
101	mumbai	123
103	Delhi	345
105	chennai	672
1	2	3

SUPPLIERS

Supp_Id	Sup_address	Contact
101	mumbai	567
101	mumbai	123
107	chennai	789
1	2	3

(int)

(varchar)

Int (type)

SQL

* SELECT (*) FROM customers UNION

SELECT (*) FROM Suppliers ;

Same, Data types Lekapothe, Em avuthadi.

Eppudu.

SELECT Address (varchar) FROM customers (varchar)

UNION

SELECT Supp_Id (int) FROM suppliers (varchar);

Output ERROR

SQL

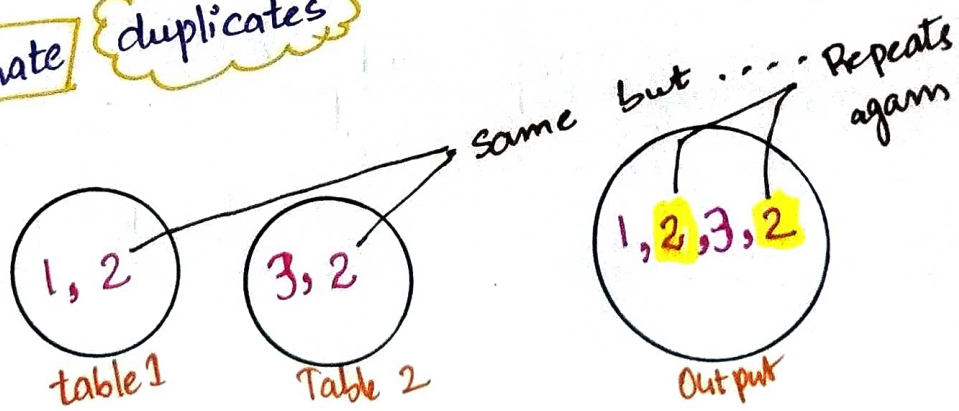
SELECT Address FROM customers

UNION

SELECT Address FROM Suppliers;

UNION ALL

* it is similar but, with one difference it doesn't eliminate duplicates



Detailed Example in SQL.

- CREATE Table Table 1 (ID (Small INT), Name (varchar (12))
- INSERT INTO Table 1 Values (1, "A")
- INSERT INTO Table 1 values (2, "B")
- INSERT INTO Table 1 Values (3, "C")

Table 1 :

ID	NAME
1	A
2	B
3	C

- CREATE Table Table 2 (ID INT, Name nvarchar(7))
- INSERT INTO Table 2 Values (1, "A")
- INSERT INTO Table 2 Values (2, "B")
- INSERT INTO Table 2 Values (3, "C")
- INSERT INTO Table 2 Values (4, "D")

Table 2 :

ID	NAME
1	A
2	B
3	C
4	D

- SELECT ID, Name FROM table 1
  UNION ALL
  SELECT ID, Name FROM table 2

  } Block of code.

Output:

ID	Name
1	A
2	B
3	C
1	A
2	B
3	C
4	D

Same

Table 1

Table 2

- SELECT ID, Name FROM Tabl
  UNION ALL
  SELECT ID, Name FROM T
  ORDER BY ID;

Output:

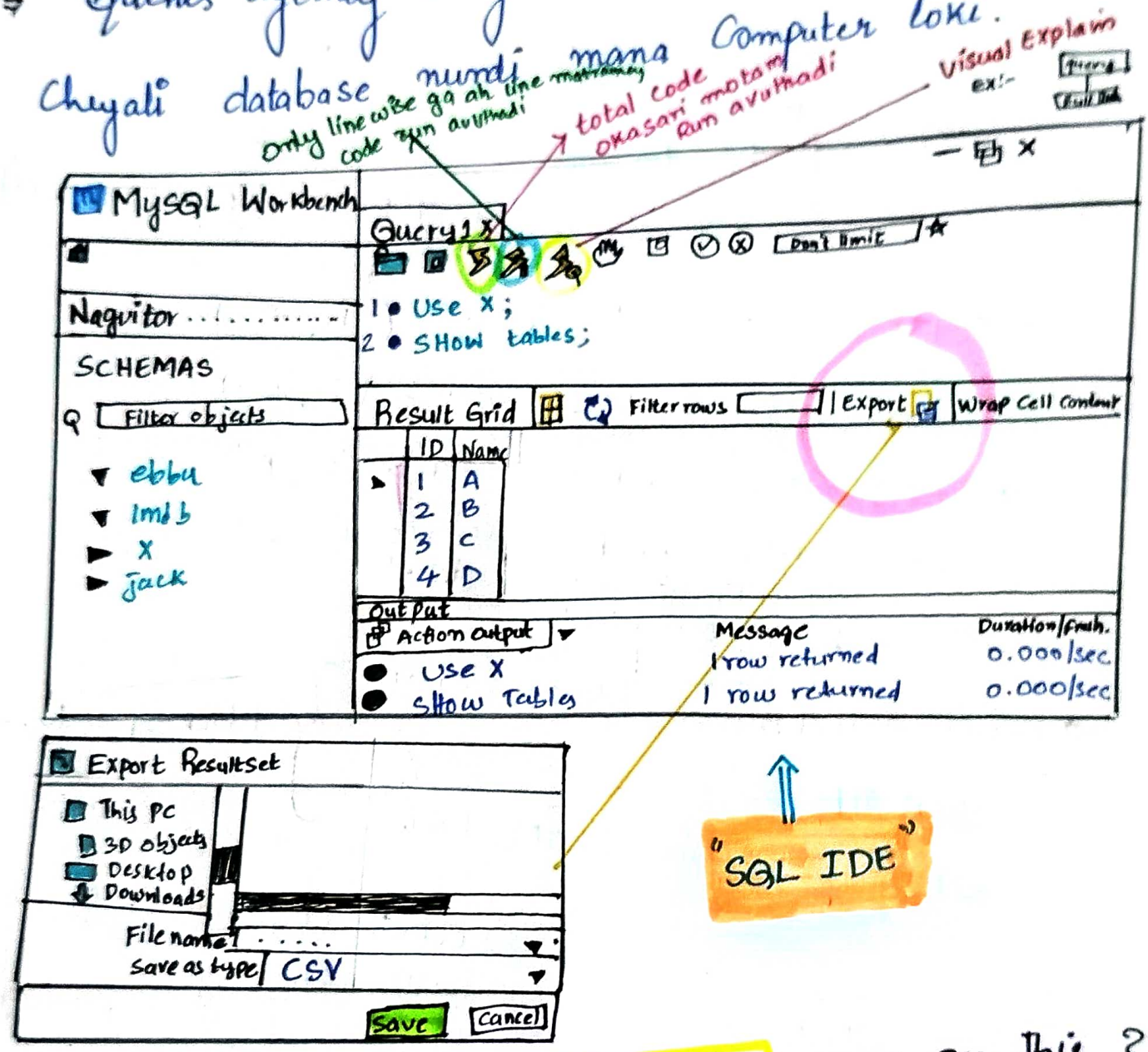
ID	NAME
1	A
1	A
2	B
2	B
3	C
3	C
4	D

it Gives
The or
wise I
and no

Dt:- 8/7/22
10:30AM

* How To Save The SQL file into CSV file ?

⇒ Queries ayethy okay ra mawa. asalu ela Save Chyali database nundi mana Computer loki.



How Engineers (or) Data Scientist work on This ?

They allows Once only One Table ? we Can't extract all tables at a time ?

So, As a Data Scientist, we Can do analysis On The data. which is Example...

Example :- After Extracting From SQL

SQL code

```
SELECT *
FROM movies;
```

↓

data

↓

Export to csv

Tabel 1 (movies) = In pandas (df 1) → dataFrame 1

Tabel 2 (movie_generes) = df2

Tabel 3 (directors) = df3

Tabel 4 (director-genre) = df4

⋮ ⋮ ⋮

Tabel 7 (roles) = df7

In pandas, (df1 - df7) we can join them into one dataFrame. and we start analysis on the data. that we extracted from SQL.

As a Engineer (or) Data Scientist

This is the process of extracting "SQL" File in csv File :

8/7/22