Floating Data Types.

digits.
2
3
:
65

powers of 10
1 (10^1)
2 (10^{64})
:
10^{64}

a) DECIMAL(p,s) → 1 to 65 ⇒ 65 digits.

→ How many values you want to store after decimal.

precision / total length

e.g. 302.04 p=5, s=2

If I know p and s, I can store the exact values in my SQL DB and put the decimal.

$1.9 - 0.9 == 1$

1.899999 0.899999

$1 - \text{epsilon} \leq 1.9 - 0.9 \leq 1 + \text{epsilon}$

epsilon ↓ 0.02

Actual	Close approximation
0.2 →	0.20001
0.01 →	0.00009
0.5 →	0.50101
0.001 →	

10^6 stocks. → 10^{-3} in each stock.

$10^6 * 10^{-3} \rightarrow 10^3$ stocks.

1000 stocks.

fee, amount, tax.

300095 → 3000.95

Rupee Paisa

Why use floating points?

→ They allow you to store a huge range of numbers

They are used

→ Where having approx values is okay.

→ You can add extra logic to handle approximation.

b) Float → 4 Bytes.

$[-3.4 * 10^{38}, 3.4 * 10^{38}]$

c) Double → 8 bytes.

$[-1.7 * 10^{308}, 1.7 * 10^{308}]$

} They store approx. values.

You use some extra logic to handle approximations if you want to use them.

Max (decimal value) in powers of 10 ⇒ 10^{64}

e.g. pi (π)

Store pi till 70 digits.

→ Can't store in decimal because it allows only 65 digits.

BOOLEAN (BOOL OR BOOLEAN)

TINY INT → 1 Byte.

BOOLEAN → True (1)
↳ false (0)

e.g. set is_deleted = TRUE where ...
OR
set is_deleted = 1 where ...

ENUMS → A set of constants.

Why use enums? → Type safety.

```
enum carTypes{
  SUV,
  SEDAN
}
```

Enums in MySQL.

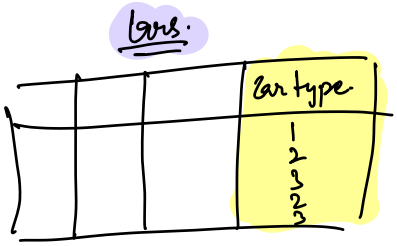
→ Values should be in the specified set of values.

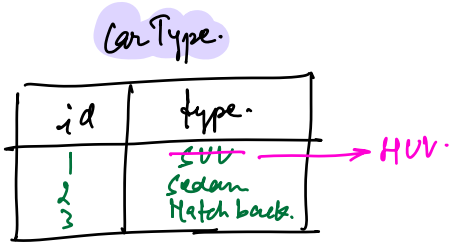
CAPS.

			carType
			"SUV"
			" HATCH"
			"SEDAN"

e.g. ENUM ('SEDAN', 'SUV', 'HATCHBACK'...)

* Practical Advice. → Don't have enum type columns in your table.
→ Have a look up table instead.





lookup table / mapping table

DATE AND TIME

1) DATE → Only going to allow you to store date. Time can't be stored here.
e.g. 7/10/2022

2) TIME → Allows you to only store time. e.g. 06:00PM

3) DATETIME → Allows you to store both date and time.
e.g. 7/10/2022 06:00PM

Timestamp / Epoch. → milliseconds since 01/01/1970 UTC

→ Don't use timestamp in MySQL.

- Timestamp has 4 Bytes in MySQL.
- In 2038, this will exhaust
- from 01/01/1970 to 01/01/2039 the timestamp will overflow.

Year 2038 problem, Epocalypse.

Blobs → Binary large objects.

TINY BLOB → 255B.

BLOB → 65KB

MEDIUM BLOB → 16MB

LONG BLOB. → 4GB.

→ Don't use blobs in databases unless you have a very good reason for it.

→ Often filestore (Aws S3) is going to be a better choice to store such large files.

Not using Blobs in dB.

→ When you store such files. You want to provide these to a CDN probably. Your CDN is not going to call dB and fetch data from there.

→ Hampers dB performance.

The General practice:-

You store metadata in dB and files in filestore.

Break:- 8:30 AM.

Normalisation

→ Technique to reduce redundancy of data in a database.

→ Helps us to create better Schema.

Problems with Redundancy.

e.g.

id	Name	gender	Phone no.	batch_id	batch_Name	instructor	
				1	Dec21	~~Rahul~~	Alok
				2	Feb22	~~Shivank~~	Alok
				1	Dec21	~~Rahul~~	Alok
				2	Feb22	Shivank	
				2	Feb22	Shivank	
				1	Dec21	Rahul	
				2	Feb22	~~Shivank~~	
NULL	NULL	NULL	NULL	3	Nov22	Alok	→ insert

1) Insertion Anamoly.

You can't create a new batch without atleast 1 student.

2) Deletion Anomoly.

Deleting other entities (like student data) you might end up deleting batch data as well.

3) Update Anamoly.

To avoid these anamolies we do:-

DB Normalisation.

→ Process to organise/structure data in a dB.

→ Reducing redundancy is the main motive here.

1NF
↓
2NF
↓
3NF
↓
BCNF (Boyce Codd NF)
↓
4NF
↓
5NF
↓
6NF

Irrelevant practically.

1NF :- Every column should have atomic values.

No multivalued Attributes.

e.g

Students.

id	name	phone no. 1	phone no. 2	phone no. 3	phone no 4
1		~	~	NULL	NULL
2				NULL	NULL
3		~	—	NULL	NULL
4		~	—		~

max (# phone numbers)

→ Sparse table

→ Mapping table

Students.

id	Name

Phone no.

id	stud_id	phno.

* (stud_id, phone no.) as PK
* Benefit:- Student data sorted. You can get phone no. of specific student faster.

2NF:- Mentor Sessions. → Information relating to mentor sessions.

PK

stud_id	mentor_id	session_datetime	feedback	mentor_Name

→ Not following 2NF.

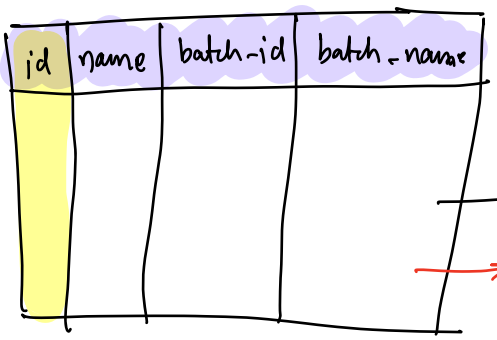
No non-prime attribute (attribute not part of PK) should depend on subset of PK.

→ All the non-prime attributes should depend on the complete PK.

3NF

→ No non-prime attribute should indirectly depend on the PK.

Students.



stud-id → batch-id → batch-name.

→ 2NF is followed here.

→ 3NF is not followed here.

Conclusion:- batch name is not directly dependent on PK
Rather it is transitively dependent.

1. Set up Workbench and SQL &
2. CRUD.
3. Joins.