```
CONSTRAINT  order-fk  FOREIGN KEY (cid) REFRENCE customer(cid)
ON DELETE  CASCADE
ON UPDATE  CASCADE
)
```

Set default :- * customer table created by same command

```
CREATE TABLE orders (
    order-id   INTEGER    PRIMARY KEY,
    cid   INTEGER,    don't write NOT NULL because cid value change into NULL after deleting.
    order-date   DATETIME   NOT NULL   DEFAULT CURRENT-TIMESTAMP,
    CONSTRAINT   order-fk   FOREIGN KEY (cid) REFRENCE customer (cid)
    ON DELETE SET NULL,
)
```

ALTER TABLE COMMAND

The Alter Table statement in SQL is used to modify the structure of an existing table. Some of the things than can be done using the Alter TABLE statement include

1. Add columns
2. Delete columns
3. Modify columns

ALTER TABLE customers ADD COLUMN password VARCHAR(255) NOT NULL,

↳ add column in existing table

ALTER TABLE customers ADD COLUMN password VARCHAR(255) NOT NULL $\boxed{\text{AFTER name}}$ → After name new col.

↳ Add column at specific place

ALTER TABLE customers

ADD COLUMN pan-number VARCHAR(255) AFTER surname,

ADD COLUMN joining-date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP

↳ Add multiple column at specific place

Delete

ALTER TABLE customers DROP COLUMN pan-number

↳ Delete existing column in a table

ALTER TABLE customer

DROP COLUMN password,

DROP COLUMN joining-date

↳ Delete multiple column

Modify

ALTER TABLE customers

MODIFY COLUMN ~~SURNAME~~ INTEGER ~~AUTO_INCREA~~ - ~~NEXT~~,

change ↳ type → INTEGER

↳ modify single or multicolumn

Editing and Deleting Constraints

1. Add → ALTER TABLE customers ADD CONSTRAINT customer-age CHECK (age>13)
2. Delete → ALTER TABLE custom DROP CONSTRAINT custome - age
3. Edit not → Not edit constrain → first delete then ~~edit~~ add constrain

Adds If you adding any constraint then firstly check column value match the constraint condition

eg:- customers

name	age
	10

→ Already have column with value and add constraint than age value is not less than 10 but already value is less than 10 then produce error, so, first change value.

Insert

→ column name

INSERT INTO Database-Name . Table_name (user_id, name, email, Password)
space ↑

Values (NULL, 'nitish', 'nitish@gmail.com', '1234')
or
VALUES

Insert without column name:

INSERT INTO Database-Name . Table-name
VALUES (NULL, 'ankit', 'ankit@gmail.com', '12345')
Space ↑

Insert in specific column

INSERT INTO Database-Name . Table-name (name, email)
space ↑
VALUES ('amit', 'amit@gmail.com')

* Order of columns is not important while inserting the data. we can also write (email, name).

Insert Multiple Values

Database   Table
INSERT INTO campus . users VALEUS

(NULL, 'rishabh', 'rishabh@gmail.com', '12345'),
(NULL, 'rohit', 'rohit@gmail.com', '12345'),
(NULL, 'rohan', 'rohan@gmail.com', ('12344')

select all → select all the columns

select all rows and columns.

$\overbrace{\text{optional write}}$

$\text{SELECT } \underbrace{\circledast}_{\text{all columns}} \text{ FROM } \underbrace{\text{campus}}_{\text{database}} . \underbrace{\text{smartphones}}_{\text{Table}} \text{ WHERE } \underbrace{1}_{\text{not applying any condition or all rows chahiye}}$

Filter cols ↓

Select some colums and all rows

$\text{SELECT } \underbrace{\text{model, price, rating}}_{\text{Column name}} \text{ FROM campus . smartphone}$

alias → Renaming cols

Rename column name

$\text{SELECT } \underbrace{\text{OS}}_{\text{column}} \text{ AS } \underbrace{\text{'operating system'}}_{\text{new name of column}}, \text{ model, } \overbrace{\text{battery\_capacity}}^{\text{oldname of column}} \text{ as } \overbrace{\text{mAH}}^{\text{new name}}$

FROM campus. smartphones

Create expression using cols

Create mathematical expression

$\text{SELECT model, sqrt(} \underline{\text{resolution\_width}} * \overset{\text{colums}}{\underline{\text{resolution\_width}}} + \underline{\text{resolution\_height}} * \underline{\text{resolution\_height}} \text{) / screen\_size AS 'ppi'}$

FROM campus. smartphones

Constants

Create new column and write value of new column

SELECT model, 'smartphone' AS 'type' from campus. smartphone

model	type
-	smartphone
-	"
-	"
.	"

- Distinct (unique) value from a column

SELECT DISTINCT (brand-name) AS 'All brands'
FROM campus. smartphones

- Distinct combination

Extract unique combination

SELECT DISTINCT brand_name, processor_name,
FROM campus . smartphones

- Filter rows where clause

SELECT $\overset{\text{all columns}}{\textcircled{*}}$. FROM campus. smartphone

WHERE brand_name = 'Samsung'
↳ select those rows which contain samsung in brand_name column.

- Between

SELECT * FROM campus. smartphones
WHERE price > '10000' AND price < '20000'

Or

SELECT * FROM campus. smartphones
WHERE price BETWEEN '10000' AND '20000'

- Order of Query Execution

From → Join → WHERE → Group by → Having → SELECT → Distinct → Order by

- IN and NOT IN

```
SELECT * FROM campus. smartphone
WHERE processor_brand = 'snapdragon' OR
processor_brand = 'exynos'
```

OR

```
SELECT * FROM campus. smartphone
WHERE processor_brand IN ('snapdragon', 'exynos')
```

row→ value present in processor_brand

```
SELECT * FROM campus. smartphone
WHERE processor_brand NOT IN ('snapdragon', 'exynos')
```

those where snapdragon and exynos not present in processor_brand

- Update

```
UPDATE campus. smartphone
SET processor_brand = 'snapdragon'
WHERE processor_brand = 'dimensity'
```

Multiple Column

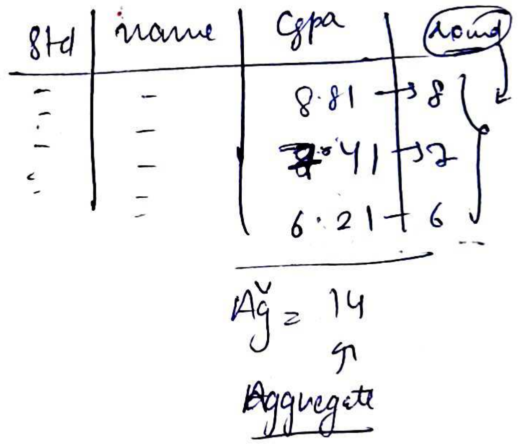
```
UPDATE campus. smartphone
SET email = 'nitish@yahoo.com', password = '123456'
WHERE name = 'nitish'
```

- Delete

DELETE FROM campus. smartphone
WHERE price > 200000.

Types of function in SQL

$$\begin{tikzcd}
 & \text{SQL Function} \arrow[dl] \arrow[dr] & \\
\boxed{\text{Built in}} \arrow[d] \arrow[dr] & & \boxed{\text{Users defined}} \\
\boxed{\text{Scaler}} \ (\text{round}) \arrow[d] & \boxed{\text{Aggregate}} \ (\text{summary}) \arrow[d] & \\
\text{Multiple value} & \text{Single value} &
\end{tikzcd}$$



Aggregate function

→ MAX / MIN

SELECT MAX(Price) from campus. smartphone
↓
MIN( )

→ AVG

SELECT AVG (Price) from campus. smartphone
WHERE brand_name = 'apple'

# Sum
SELECT sum (price) FROM campus. smartphone
WHERE brand-name = 'apple'

# Count
SELECT COUNT(*) from campus. smartphone
WHERE brand_name = 'apple'

# Count (DISTINCT)
SELECT COUNT (DISTINCT ('processo-brand')) FROM campus. Smartpho

# STD (Standard Deviation)
SELECT STD(screen-size) FROM campus. Smartphone.

Variance
SELET VARIANCE (screen-size) from campus. Smart.

Scaler Function

→ ABS → all the values are positive
SELECT ABS (price- 100000) as 'temp' from campus. Smartphon

→ Round
SELECT model,
ROUND (Price, 2) → 2 decimals 28.10421 = 28.10
FROM campus, Smartphone

convert into single digit = 28.10421 = 28
round ↑

→ CEIL / FLOOR

CEIL ↓ next integer

FLOOR ↳ befor integer

4.9 → 5   4.9 → 4

4.1 → 5   4.1 → 4

SELECT CEIL(screen-size) FROM Campus. Smartphone

↳ FLOOR

Sorting

SELECT * FROM campus . smartphones WHERE brand_name = 'Samsung'

ORDER BY screen_size DESC

screen_size ↲ column

DESC ↳ descending

Set limit of rows

SELECT * FROM campus. smartphones WHERE brand_name = 'Samsung'

ORDER BY screen_size DESC LIMIT 5

↳ first five lines show
↳ row

Select only 2 row

SELECT model, battery-capacity FROM campus. smartphones
ORDER BY battery-capacity DESC LIMIT 1,1

row → skip → 0
① → start

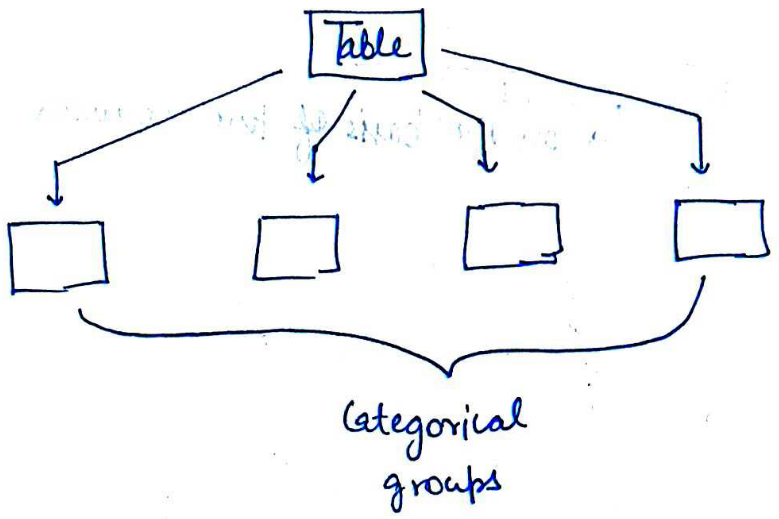
next row print

0-4 skip
↳ 5,5
↳ 5-9
↳ print

Sort one the basis of Two column

```
SELECT brand-name, price FROM campus.smartphones
ORDER BY brand-name ASC, price ASC
```

Grouping Data



Groupby according to brand-name of phone and count of phones.

```
SELECT brand-name, COUNT(*) AS 'num-phones'
FROM campus. smartphones
GROUP BY brand-name ORDER BY DESC
                        ↳ higher count
```

Group by multiple column. On the basis of single column

```
SELECT brand-name,
COUNT (*) AS 'num-phones',        → 2 columns
ROUND (AVG (price)) AS 'avg-price'
FROM campus. smartphones
GROUP BY brand-name → single column base
ORDER BY num-phone DESC LIMIT 5
```

Group by of multiple column on the basis of multiple column

```
SELECT brand_name,
processor-name,
COUNT (*) AS 'num_phones',
ROUND (AVG (primary_cam_reso) AS 'avg-cam-reso'
FROM campus. smartphones
GROUP BY brand-name, processor_name
```

↳ on the basis of two column

Having → filtering on groups and WHERE → Filtering on normals

Ques:- Find the avg rating of smartphones brands which have more than 20 phones.

```
SELECT brand_name,
COUNT (*) AS 'count',
AVG (~~price~~ rating) AS 'avg-~~price~~ rating'
FROM campus. smartphones
GROUP BY brand-name
HAVING count > 20
```

Ques → Find the top 3 brands with the highest avg ram that have a refresh rate of at least 90Hz and Fast charging available and don't consider brands which have less than 10 phones.

```
SELECT brand-name,
COUNT(*) AS 'count',
ROUND(AVG(ram-capacity)) AS 'avg-ram'
FROM campus. smartphones
WHERE refresh_rate >= 90 AND fast-charging-available=1
GROUP BY brand name
HAVING count > 10
ORDER BY avg-ram DESC LIMIT 3
```

Ques:- Find the avg price of all the phone brands with avg rating > 70 and num phones more than 10 among all 5g enabled phones.

```
SELECT brand-name,
AVG(price) AS 'avg-price'
FROM campus. smartphones
WHERE has-5g = 'True'
GROUP BY brand-name
HAVING AVG(rating) > 70 AND COUNT(*) >10
```

Practice

1. Find the top 5 batsmas in IPL
2. Find the 2nd highest 6 hitter in IPL
3. Find virat kohli's performance against all IPL teams
4. Find to 10 batsman with centuries in IPL
5. Find the top 5 batsmen with highest strike rate who have played a min of 1000 balls