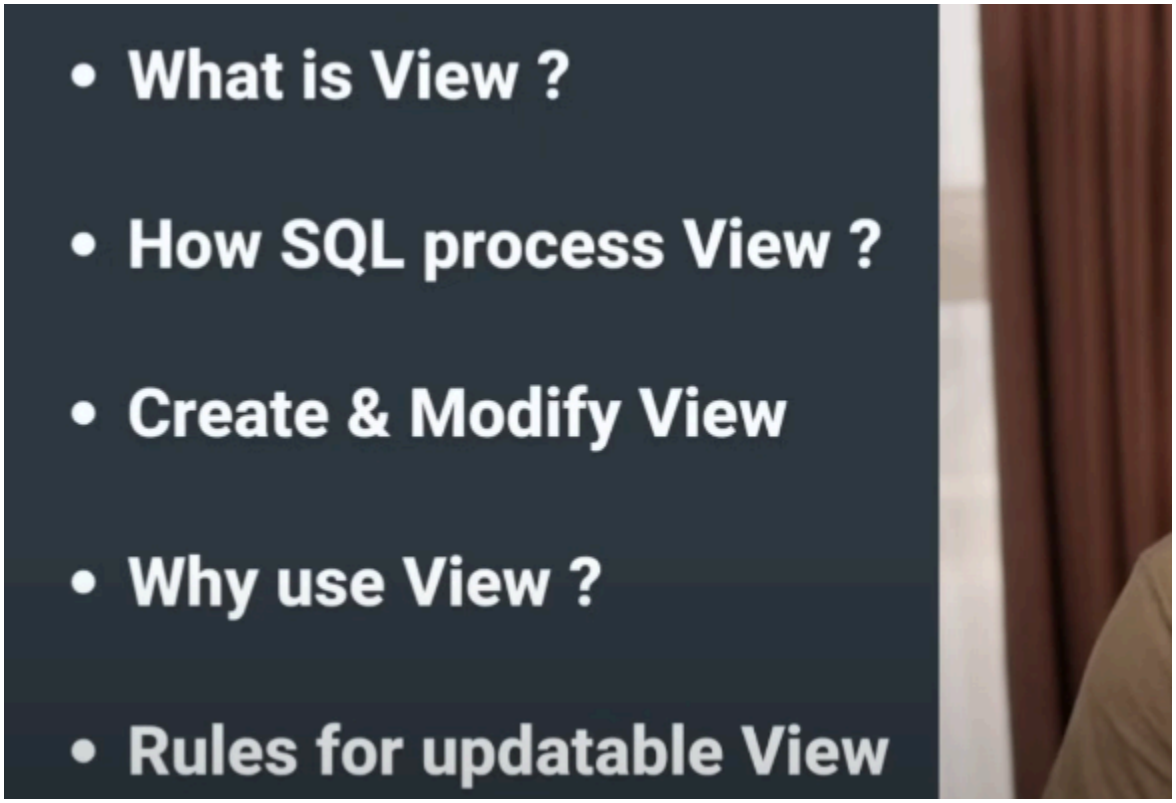
Views



View is a Database Object

View is created over an SQL Query

Notifications

View does NOT store any data

View is like a virtual table

CLICK HERE TO

```
-- What is the main purpose of using a view / advantages of views.
1) Security
2) To simplfiy complex sql queries.
```

```
create role james
login
password 'james';
```

```
1  select * from order_summary;
2
```

Data Output | Explain | Messages | Notifications

```
ERROR:  permission denied for view order_summary
SQL state: 42501
```

```
grant select on order_summary to james;
```

1) Security
- By hiding the query used to generate the view

2) Simplify complex SQL queires
- Sharing a View is better than sharing complex query
- Avoid re-writing same complex query multiple times

```
6  create or replace view order_summary
7  as
8  select o.ord_id, o.date, p.prod_name, c.cust_name
9  , (p.price * o.quantity) - ((p.price * o.quantity) * disc_percent::float/100) as cost
10 from tb_customer_data c
11 join tb_order_details o on o.cust_id = c.cust_id
12 join tb_product_info p on p.prod_id = o.prod_id;
13
14
15
```

Query returned successfully in 54 msec.
Query returned successfully in 47 msec.
Query returned successfully in 52 msec.
Query returned successfully in 51 msec.
Query returned successfully in 45 msec.

Data Output | Explain | Messages | Notifications

```
CREATE VIEW

Query returned successfully in 56 msec.
```

There are rules when we use replace
- Can not change the order of columns
- Can not change data types
- Can not change column name
- Can add new columns at the end
- Can add join in tables

Then how can we change structure of view - by using Alter

```
18
19 alter view order_summary rename column date to order_date;
20
21
22
```

Data Output | Explain | Messages | Notifications

```
ALTER VIEW

Query returned successfully in 49 msec.
```

Can rename view also

```
21 alter view order_summary rename to order_summary_2;
22
```

Can drop view

```
23  drop view order_summary_2;
24
```

```
26  create view expensive_products
27  as
28  select * from tb_product_info where price > 1000;
29
30  select * from expensive_products;
31  select * from tb_product_info;
32
33  alter table tb_product_info add column prod_config varchar(100)
34
35
36
```

Data Output | Explain | Messages | Notifications

	prod_id [PK] character varying (10)	prod_name character varying (50)	brand character varying (50)	price integer	prod_config character varying (100)
1	P1	Samsung S22	Samsung	800	[null]
2	P2	Google Pixel 6 Pro	Google	900	[null]
3	P3	Sony Bravia TV	Sony	600	[null]

Note:

- View store structure of table
- View always show latest data

```sql
-- Updatable views
1) Views should be created using 1 table/view only

create or replace view expensive_products
as
select * from tb_product_info where price > 1000;

select * from expensive_products;
select * from tb_product_info;

update expensive_products
set prod_name = 'Airpods Pro', brand = 'Apple'
where prod_id = 'P10';
```

Data Output | Explain | Messages | Notifications

	prod_id [PK] character varying (10)	prod_name character varying (50)	brand character varying (50)	price integer	prod_config character varying (100)
1	P1	Samsung S22	Samsung	800	[null]
2	P2	Google Pixel 6 Pro	Google	900	[null]
3	P3	Sony Bravia TV	Sony	600	[null]
4	P4	Dell XPS 17	Dell	2000	[null]
5	P5	iPhone 13	Apple	800	[null]
6	P6	Macbook Pro 16	Apple	5000	[null]
7	P10	Airpods Pro	Apple	1200	[null]

```sql
update order_summary
set cost = 10
where ord_id = 1;
```

Data Output | Explain | Messages | Notifications

```
ERROR:  cannot update view "order_summary"
DETAIL:  Views that do not select from a single table or view are not automatically updatable.
HINT:  To enable updating the view, provide an INSTEAD OF UPDATE trigger or an unconditional ON UPDATE DO INSTEAD rule.
SQL state: 55000
```

```
60  2) cannot have DISTINCT clause.
```

```
68  update expensive_products
69  set prod_name = 'Airpods Pro 2', brand = 'Apple'
70  where prod_id = 'P10';
71
72
73
```

Data Output | Explain | Messages | Notifications

```
ERROR:  cannot update view "expensive_products"
DETAIL:  Views containing DISTINCT are not automatically updatable.
HINT:  To enable updating the view, provide an INSTEAD OF UPDATE trigger or an unconditional ON UPDATE DO INSTEAD rule.
SQL state: 55000
```

```
73  3) if query contains GROUP BY then cannpot update such views.
```

```
88  4) if query contains WITH clause then cannot update such views.
89
90  5) If query contains window functions then cannot update such views.
```

```
94  -- WITH CHECK OPTION
95
96  create or replace view apple_products
97  as
98  SELECT * from tb_product_info where brand = 'Apple'
99  with check option ;
100
101 insert into apple_products
102 values ('P20', 'Note 20', 'Samsung', 2500, null);
103
104 select * from tb_product_info;
105 select * from apple_products;
106
107
```

Data Output | Explain | Messages | Notifications

```
CREATE VIEW

Query returned successfully in 50 msec.
```

```
101 insert into apple_products
102 values ('P22', 'Note 22', 'Samsung', 2500, null);
103
104 select * from tb_product_info;
105 select * from apple_products;
106
107
```

Data Output | Explain | Messages | Notifications

```
ERROR:  new row violates check option for view "apple_products"
DETAIL:  Failing row contains (P22, Note 22, Samsung, 2500, null).
SQL state: 44000
```

Normal vs Materialized Views

Feature	Normal View	Materialized View
Storage	Does not store data physically	Stores data physically
Data Freshness	Always up-to-date	Can become stale; requires refresh
Performance	Slower for complex queries (depends on base table)	Faster for read operations; data is precomputed
Usage	Simplifies complex queries, provides abstraction	Improves performance for expensive queries, stores aggregated data
Refresh	No refresh needed	Needs manual or automatic refresh
Data Access	Fetches data dynamically from base tables	Retrieves precomputed data

Materialized View

```
1  create table random_tab (id int, val decimal);
2
3  insert into random_tab
4  select 1, random() from generate_series(1, 10000000);
5
6  insert into random_tab
7  select 2, random() from generate_series(1, 10000000);
8
9  select id, avg(val), count(*)
10 from random_tab
11 group by id;
12
13 create materialized view mv_random_tab
14 as
15
```

Data Output | Explain | Messages | Notifications

	id integer	avg numeric	count bigint
1	1	0.5000038332533442522356	10000000
2	2	0.4998689635719952220993O	10000000

```
21 delete from random_tab where id = 1;
22
23 refresh materialized view mv_random_tab;
```