5————————————

```
SELECT
  id,
  (SELECT name FROM users WHERE users.id=private_videos_orders.customer_id) AS customer_name,
   (SELECT name FROM users WHERE users.id=private_videos_orders.owner_id) AS owner_name,
    (SELECT
        (SELECT name FROM payment_systems WHERE payment_systems.id=payments.payment_system_id)
     FROM payments WHERE payments.order_id=private_videos_orders.id) AS payment_system,
          order_date
FROM private_videos_orders
ORDER BY order_date DESC LIMIT 20;
```

————————————————————

```
SELECT  order_id,
  payment_systems.name AS payment_system,
  payment_statuses.name AS status,
  private_video_id,
  size
FROM payments
  JOIN private_videos_orders
    ON payments.order_id = private_videos_orders.id
  JOIN payment_statuses
        ON payment_statuses.id = payments.status
  JOIN payment_systems
        ON payment_systems.id = payments.payment_system_id
  LEFT JOIN videos
    ON videos.id = private_videos_orders.private_video_id
```

```
ORDER BY size DESC
LIMIT 10;
```

6————————————————-

```
SELECT order_id,
status,
price,
private_video_id
FROM payments
JOIN private_videos_orders
ON order_id = private_videos_orders.id
```

_____

```
SELECT videos.id,
title,
category_public AS categroy,
name
FROM videos
JOIN users
ON author_id = users.id
JOIN  categories
on category =  categories.id
LIMIT 10;
```

7————————————

```
CREATE OR REPLACE VIEW video_view AS
SELECT videos.id, videos.size, users.name
```

```
  FROM videos
    LEFT JOIN users
      ON videos.author_id = users.id
  WHERE videos.size > 23;
SELECT * FROM video_view LIMIT 5;
—
CREATE OR REPLACE VIEW video_orders AS
SELECT private_videos_orders.id, private_videos_orders.order_date, payments.status,
payments.price
  FROM private_videos_orders
    LEFT JOIN payments
      ON private_videos_orders.id = payments.order_id
  WHERE private_videos_orders.order_date > (current_timestamp - interval '3 month');
SELECT * FROM video_orders;
```

8————————————————————-

```
CREATE FUNCTION video_customers(user_id INTEGER)
RETURNS INTEGER AS
$$
SELECT customer_id
FROM private_videos_orders
WHERE owner_id = user_id
GROUP BY customer_id
ORDER BY count(*) DESC
LIMIT 5;
$$
LANGUAGE sql;
```

```
SELECT video_customers(55);
```

9————————————

```
CREATE OR REPLACE FUNCTION video_cheak_before_delete_trigger()
RETURNS TRIGGER AS
$$
BEGIN
  IF OLD.category = 1 THEN
  RAISE EXCEPTION 'Active orders are found. Video can not be removed.';
    END IF;
  RETURN NEW;
END;
$$
LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS video_cheak_before_delete ON videos;
CREATE TRIGGER video_cheak_before_delete
BEFORE DELETE ON videos
FOR EACH ROW
EXECUTE FUNCTION video_cheak_before_delete_trigger();

SELECT * FROM videos

DELETE FROM videos
WHERE id = 99;
ERROR:  Active orders are found. Video can not be removed.
```

10———————————

EXPLAIN ANALYZE SELECT order_id,

status,

price,

private_video_id

FROM payments

JOIN private_videos_orders

ON order_id = private_videos_orders.id

WHERE status = 3

Query Query History

```
1 EXPLAIN ANALYZE SELECT order_id,
2 status,
3 price,
4 private_video_id
5 FROM payments
6 JOIN private_videos_orders
7 ON order_id = private_videos_orders.id
8 WHERE status = 3
```

Data output Messages Notifications

	QUERY PLAN text
1	Hash Join (cost=2.59..5.61 rows=27 width=16) (actual time=0.031..0.048 rows=27 loops=1)
2	Hash Cond: (private_videos_orders.id = payments.order_id)
3	-> Seq Scan on private_videos_orders (cost=0.00..2.00 rows=100 width=8) (actual time=0.005..0.010 rows=100 loops...
4	-> Hash (cost=2.25..2.25 rows=27 width=12) (actual time=0.021..0.021 rows=27 loops=1)
5	Buckets: 1024 Batches: 1 Memory Usage: 10kB
6	-> Seq Scan on payments (cost=0.00..2.25 rows=27 width=12) (actual time=0.005..0.014 rows=27 loops=1)
7	Filter: (status = 3)
8	Rows Removed by Filter: 73
9	Planning Time: 0.140 ms
10	Execution Time: 0.067 ms

Индекс

CREATE INDEX payments_status_fk ON payments (status);

CREATE INDEX payments_order_id_fk ON payments (order_id);