MATERIALIZED VIEWS

```
\d orders

                Table "public.orders"
   Column    |  Type   | Collation | Nullable | Default
-------------+---------+-----------+----------+---------
 order_id    | integer |           |          |
 customer_id | integer |           |          |
 amount      | numeric |           |          |

SELECT count(*) FROM orders;

  count
----------
 20000000
(1 row)

SELECT * FROM orders LIMIT 10;

 order_id | customer_id | amount
----------+-------------+--------
        1 |           2 | 188.12
        2 |           2 |  95.80
        3 |           2 | 252.20
        4 |           3 | 383.40
        5 |           3 | 297.55
        6 |           1 | 172.34
        7 |           2 | 180.04
        8 |           3 | 270.27
        9 |           2 | 291.12
       10 |           3 | 435.66
(10 rows)
```

```
SELECT customer_id, count(1), avg(amount) FROM orders GROUP BY customer_id;

 customer_id |  count  |         avg
-------------+---------+---------------------
           1 | 6671419 | 250.0531631216687185
           2 | 6665436 | 249.9877122486811065
           3 | 6663145 | 250.0006444809470603
(3 rows)

CREATE VIEW v_cust AS SELECT customer_id, count(1), avg(amount) FROM orders GROUP BY customer_id;
CREATE VIEW

SELECT * FROM v_cust;

 customer_id |  count  |         avg
-------------+---------+---------------------
           1 | 6671419 | 250.0531631216687185
           2 | 6665436 | 249.9877122486811065
           3 | 6663145 | 250.0006444809470603
(3 rows)
```

```
EXPLAIN ANALYZE SELECT * FROM v_cust;

                                                                        QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------
 Finalize GroupAggregate  (cost=254943.62..254944.41 rows=3 width=44) (actual time=948.322..949.728 rows=3 loops=1)
   Group Key: customer_id
   ->  Gather Merge  (cost=254943.62..254944.32 rows=6 width=44) (actual time=948.313..949.718 rows=9 loops=1)
         Workers Planned: 2
         Workers Launched: 2
         ->  Sort  (cost=253943.59..253943.60 rows=3 width=44) (actual time=933.816..933.816 rows=3 loops=3)
               Sort Key: customer_id
               Sort Method: quicksort  Memory: 25kB
               Worker 0:  Sort Method: quicksort  Memory: 25kB
               Worker 1:  Sort Method: quicksort  Memory: 25kB
               ->  Partial HashAggregate  (cost=253943.53..253943.57 rows=3 width=44) (actual time=933.784..933.785 rows=3 loops=3)
                     Group Key: customer_id
                     Batches: 1  Memory Usage: 24kB
                     Worker 0:  Batches: 1  Memory Usage: 24kB
                     Worker 1:  Batches: 1  Memory Usage: 24kB
                     ->  Parallel Seq Scan on orders  (cost=0.00..191443.02 rows=8333402 width=10) (actual time=0.079..283.788 rows=6666667 loops=3)
 Planning Time: 0.210 ms
 Execution Time: 949.820 ms
(18 rows)
```

```
CREATE MATERIALIZED VIEW mv_cust AS SELECT customer_id, count(1), avg(amount) FROM orders GROUP BY customer_id;
SELECT 3

\d
               List of relations
 Schema |  Name   |       Type        |  Owner
--------+---------+-------------------+----------
 public | mv_cust | materialized view | postgres
 public | orders  | table             | postgres
 public | v_cust  | view              | postgres
(3 rows)

SELECT * FROM pg_matviews WHERE matviewname = 'mv_cust';

 schemaname | matviewname | matviewowner | tablespace | hasindexes | ispopulated |       definition
------------+-------------+--------------+------------+------------+-------------+-------------------------
 public     | mv_cust     | postgres     |            | f          | t           |  SELECT customer_id,   +
            |             |              |            |            |             |     count(1) AS count, +
            |             |              |            |            |             |     avg(amount) AS avg +
            |             |              |            |            |             |    FROM orders         +
            |             |              |            |            |             |   GROUP BY customer_id;
(1 row)

SELECT * FROM mv_cust;

 customer_id |  count  |         avg
-------------+---------+---------------------
           1 | 6671419 | 250.0531631216687185
           2 | 6665436 | 249.9877122486811065
           3 | 6663145 | 250.0006444809470603
(3 rows)
```

```
EXPLAIN ANALYZE SELECT * FROM v_cust;

                                                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------------
 Finalize GroupAggregate  (cost=254943.62..254944.41 rows=3 width=44) (actual time=948.322..949.728 rows=3 loops=1)
   Group Key: customer_id
   ->  Gather Merge  (cost=254943.62..254944.32 rows=6 width=44) (actual time=948.313..949.718 rows=9 loops=1)
         Workers Planned: 2
         Workers Launched: 2
         ->  Sort  (cost=253943.59..253943.60 rows=3 width=44) (actual time=933.816..933.816 rows=3 loops=3)
               Sort Key: customer_id
               Sort Method: quicksort  Memory: 25kB
               Worker 0:  Sort Method: quicksort  Memory: 25kB
               Worker 1:  Sort Method: quicksort  Memory: 25kB
               ->  Partial HashAggregate  (cost=253943.53..253943.57 rows=3 width=44) (actual time=933.784..933.785 rows=3 loops=3)
                     Group Key: customer_id
                     Batches: 1  Memory Usage: 24kB
                     Worker 0:  Batches: 1  Memory Usage: 24kB
                     Worker 1:  Batches: 1  Memory Usage: 24kB
                     ->  Parallel Seq Scan on orders  (cost=0.00..191443.02 rows=8333402 width=10) (actual time=0.079..283.788 rows=6666667 loops=3)
 Planning Time: 0.210 ms
 Execution Time: 949.820 ms
(18 rows)

EXPLAIN ANALYZE SELECT * FROM mv_cust;

                                               QUERY PLAN
-----------------------------------------------------------------------------------------------------
 Seq Scan on mv_cust  (cost=0.00..21.30 rows=1130 width=44) (actual time=0.032..0.036 rows=3 loops=1)
 Planning Time: 0.158 ms
 Execution Time: 0.069 ms
(3 rows)
```

```
SELECT * FROM v_cust;

 customer_id |  count  |         avg
-------------+---------+---------------------
           1 | 6671419 | 250.0531631216687185
           2 | 6665436 | 249.9877122486811065
           3 | 6663145 | 250.0006444809470603
(3 rows)

SELECT * FROM mv_cust;

 customer_id |  count  |         avg
-------------+---------+---------------------
           1 | 6671419 | 250.0531631216687185
           2 | 6665436 | 249.9877122486811065
           3 | 6663145 | 250.0006444809470603
(3 rows)

INSERT INTO orders VALUES (20000001, 4, 500);
INSERT 0 1

SELECT customer_id, count(1), avg(amount) FROM orders GROUP BY customer_id;

 customer_id |  count  |         avg
-------------+---------+---------------------
           1 | 6671419 | 250.0531631216687185
           2 | 6665436 | 249.9877122486811065
           3 | 6663145 | 250.0006444809470603
           4 |       1 | 500.0000000000000000
(4 rows)
```

```
SELECT * FROM v_cust;

 customer_id |  count  |         avg
-------------+---------+----------------------
           1 | 6671419 | 250.0531631216687185
           2 | 6665436 | 249.9877122486811065
           3 | 6663145 | 250.0006444809470603
           4 |       1 | 500.0000000000000000
(4 rows)

SELECT * FROM mv_cust;

 customer_id |  count  |         avg
-------------+---------+----------------------
           1 | 6671419 | 250.0531631216687185
           2 | 6665436 | 249.9877122486811065
           3 | 6663145 | 250.0006444809470603
(3 rows)

REFRESH MATERIALIZED VIEW mv_cust;
REFRESH MATERIALIZED VIEW

SELECT * FROM mv_cust;

 customer_id |  count  |         avg
-------------+---------+----------------------
           1 | 6671419 | 250.0531631216687185
           2 | 6665436 | 249.9877122486811065
           3 | 6663145 | 250.0006444809470603
           4 |       1 | 500.0000000000000000
(4 rows)
```

Refresh Frequency

- **Auto refresh** - every time the data changes
  - Materialized view always up-to-date
  - Most expensive
- **Fixed frequency** - fixed time interval
  - Simple to implement
  - Tricky to figure out the right frequency
- **Variable Frequency** - after certain number of data writes
  - Better at adjusting frequency
  - More complex to implement

	View	Materialized Views
View result	Computed each time the view is used	Pre-computed and stored in the database. Refreshed automatically or on-demand.
Data freshness	Always up-to-date	May not be up-to-date. Depends on refresh frequency.
Speed to retrieve view results	Slow	Fast
Extra Storage	No	Yes

When to use Materialized Views

- Complex query with small result set
- Data is read only or very few updates
- Don’t care if results are slightly out of date