Question 1 :

```
DROP TABLE if exists  TIME CASCADE;
CREATE TABLE time (
     TimeId serial primary key,
     OrderDate date NOT NULL,
     DayOfWeek character(10) NOT NULL,
     Month character(10) NOT NULL,
     Year integer NOT NULL
 );

INSERT INTO time ( OrderDate, DayOfWeek, Month ,Year )
SELECT  DISTINCT cust_order.orderdate ,
  to_char(   cust_order.orderdate  ,'Day') as DayOfWeeek  , to_char( cust_order.orderdate
,'Month' ) as Month,
EXTRACT(Year FROM  cust_order.orderdate ) as Year
FROM  cust_order order by orderdate  asc ;

SELECT COUNT(*) FROM TIME;
```

```
drop  materialized view if exists Sales CASCADE ;
CREATE materialized view Sales as
select customer.CustomerId as  CustomerId ,TIME.TimeId as TimeId   , book.isbn AS  ISBN ,
SUM(order_detail.quantity * book.price)  AS Amnt
from book NATURAL join order_detail NATURAL JOIN cust_order NATURAL JOIN customer
NATURAL JOIN TIME
group by  customer.CustomerId  ,  TIME.TimeId  ,book.isbn
order by  customer.CustomerId , TIME.TimeId   ,  book.isbn  ;

select COUNT(*) from Sales;
```

Question 2

```
create materialized view avg_amnt_view as select customerid,
avg(amnt) as avg_amnt from sales group by customerid;
select avg(avg_amnt) from avg_amnt_view;
```

```
select avg(amnt) from sales;


wusong3=> create materialized view avg_amnt_view as select customerid,
wusong3-> avg(amnt) as avg_amnt from sales group by customerid;
SELECT 104
wusong3=> select avg(avg_amnt) from avg_amnt_view;
        avg
---------------------
 202.9588687852809865
(1 row)

wusong3=> select avg(amnt) from sales;
        avg
---------------------
 161.3691588785046729
(1 row)
```

The second one is the correct answer because average of average is ignoring the weight of each column.

Question 3 .
a)

```
SELECT SUM (AMNT) AS money, customer.customerid , customer.l_name  , customer.f_name
from sales  NATURAL join customer
group by customer.customerid
 ORDER BY money DESC limit 5;


 wusong3=> SELECT SUM (AMNT) AS money, customer.customerid , customer.l_name  ,
customer.f_name   from sales  NATURAL join customer
 wusong3-> group by customer.customerid , customer.l_name  , customer.f_name
 wusong3->  ORDER BY money DESC limit 5;
   money   | customerid |       l_name        |       f_name
 ----------+------------+---------------------+---------------------
  17810.00 |          1 | Jacson              | Kirk
  14100.00 |          3 | Andree              | Peter
  11780.00 |         14 | Anslow              | Craig
   7145.00 |          2 | Leow                | May-N
   6095.00 |         79 | Liang               | Jiajun
 (5 rows)
```

rool up and rank are used.

b)

rank is used .

```
drop  materialized view if exists BestCS CASCADE;
create materialized view BestCS AS
  SELECT SUM (AMNT) AS money, customer.customerid AS CID, customer.l_name  ,
customer.f_name   from sales  NATURAL join customer
  group by customer.customerid  ,customer.l_name  , customer.f_name
  ORDER BY money DESC limit 1 ;



drop  materialized view if exists BestCSorders CASCADE;
create materialized view BestCSorders AS
  SELECT  order_detail.orderid, sum(order_detail.quantity * book.price) AS BST
    FROM order_detail NATURAL JOIN book NATURAL JOIN cust_order NATURAL JOIN
customer
    WHERE cust_order.customerid = (SELECT CID FROM BestCS )
    GROUP BY orderid;


drop  materialized view if exists ord_avg_amntV CASCADE;
create materialized view ord_avg_amntV AS
  select avg(costForEachOrder.oa) as ord_avg_amnt from
  (SELECT order_detail.orderid,  sum(order_detail.quantity * book.price) AS oa
  FROM order_detail NATURAL JOIN book
  GROUP BY orderid) as costForEachOrder;


drop  materialized view if exists numerator CASCADE ;
create materialized view numerator AS
SELECT COUNT(*) FROM BestCSorders ,  ord_avg_amntV  WHERE BST >
ord_avg_amntV.ord_avg_amnt   ;


drop  materialized view if exists denominator  CASCADE;
create materialized view denominator AS
```

```
SELECT  COUNT(cust_order.orderid) as no_of_ord from customer NATURAL join cust_order
where customer.customerid =  (SELECT CID FROM BestCS ) ;

drop  materialized view if exists division  CASCADE;
create materialized view division AS
SELECT  COUNT as numerator,  no_of_ord as denominator from numerator  NATURAL join
denominator ;


select numerator/denominator::float  as perc_of_ord from division ;


wusong3=> select numerator/denominator::float  as perc_of_ord from division ;
    perc_of_ord
-------------------
 0.714285714285714
(1 row)
```

since perc_of_ord is 71% , we estimate that "e that the best buyer has issued a greater (than average) to medium number of orders with greater (than average) amounts of money,"

Question 4 .
a)

```
drop  materialized view if exists View1 CASCADE ;
CREATE MATERIALIZED VIEW View1 AS
SELECT c.CustomerId, F_Name, L_Name, District, TimeId,
DayOfWeek, ISBN, Amnt
FROM Sales NATURAL JOIN Customer c NATURAL JOIN Time;

drop  materialized view if exists View2 CASCADE;
CREATE MATERIALIZED VIEW View2 AS
SELECT c.CustomerId, F_Name, L_Name, Year, SUM(Amnt)
FROM Sales NATURAL JOIN Customer c NATURAL JOIN Time
GROUP BY c.CustomerId, F_Name, L_Name, Year;


--1. The “Book Orders Database”
EXPLAIN ANALYZE
```

```
 SELECT SUM (AMNT) AS money, customer.customerid , customer.l_name  , customer.f_name
from
(select customer.CustomerId as  CustomerId ,TIME.TimeId as TimeId   , book.isbn AS  ISBN ,
SUM(order_detail.quantity * book.price)  AS Amnt
from book NATURAL join order_detail NATURAL JOIN cust_order NATURAL JOIN customer
NATURAL JOIN TIME
group by  customer.CustomerId  ,  TIME.TimeId  ,book.isbn
order by  customer.CustomerId , TIME.TimeId   ,  book.isbn  ) as sales
NATURAL join customer
group by customer.customerid , customer.l_name  , customer.f_name
 ORDER BY money DESC limit 5;
VACUUM (ANALYZE) ;

wusong3=> EXPLAIN ANALYZE
wusong3->  SELECT SUM (AMNT) AS money, customer.customerid , customer.l_name  ,
customer.f_name   from
wusong3-> (select customer.CustomerId as  CustomerId ,TIME.TimeId as TimeId   , book.isbn
AS  ISBN ,  SUM(order_detail.quantity * book.price)  AS Amnt
wusong3(> from book NATURAL join order_detail NATURAL JOIN cust_order NATURAL JOIN
customer NATURAL JOIN TIME
wusong3(> group by  customer.CustomerId  ,  TIME.TimeId  ,book.isbn
wusong3(> order by  customer.CustomerId , TIME.TimeId   ,  book.isbn  ) as sales
wusong3-> NATURAL join customer
wusong3-> group by customer.customerid , customer.l_name  , customer.f_name
wusong3->  ORDER BY money DESC limit 5;
                                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------
 Limit  (cost=197.83..197.84 rows=5 width=78) (actual time=24.364..24.380 rows=5 loops=1)
   ->  Sort  (cost=197.83..198.12 rows=118 width=78) (actual time=24.359..24.364 rows=5
loops=1)
         Sort Key: (sum((sum(((order_detail.quantity)::numeric * book.price)))))
         Sort Method: top-N heapsort  Memory: 25kB
         ->  HashAggregate  (cost=194.39..195.87 rows=118 width=78) (actual
time=23.991..24.146 rows=104 loops=1)
               Group Key: customer.customerid, customer.l_name, customer.f_name
               ->  Hash Join  (cost=133.29..187.90 rows=649 width=78) (actual time=12.429..21.238
rows=1070 loops=1)
                     Hash Cond: (customer_1.customerid = customer.customerid)
                     ->  GroupAggregate  (cost=128.63..161.63 rows=1100 width=19) (actual
time=12.060..17.896 rows=1070 loops=1)
                           Group Key: customer_1.customerid, "time".timeid, book.isbn
                           ->  Sort  (cost=128.63..131.38 rows=1100 width=19) (actual
time=12.033..13.433 rows=1100 loops=1)
```

```
                        Sort Key: customer_1.customerid, "time".timeid, book.isbn
                        Sort Method: quicksort  Memory: 134kB
                        ->  Hash Join  (cost=23.82..73.06 rows=1100 width=19) (actual
time=2.594..9.759 rows=1100 loops=1)
                              Hash Cond: (order_detail.isbn = book.isbn)
                              ->  Hash Join  (cost=22.55..56.67 rows=1100 width=14) (actual
time=2.541..6.918 rows=1100 loops=1)
                                    Hash Cond: (order_detail.orderid = cust_order.orderid)
                                    ->  Seq Scan on order_detail  (cost=0.00..19.00 rows=1100
width=10) (actual time=0.004..1.330 rows=1100 loops=1)
                                    ->  Hash  (cost=19.77..19.77 rows=222 width=12) (actual
time=2.524..2.524 rows=222 loops=1)
                                          Buckets: 1024  Batches: 1  Memory Usage: 10kB
                                          ->  Hash Join  (cost=9.45..19.77 rows=222 width=12) (actual
time=0.709..2.191 rows=222 loops=1)
                                                Hash Cond: (cust_order.orderdate = "time".orderdate)
                                                ->  Hash Join  (cost=4.65..11.93 rows=222 width=12)
(actual time=0.345..1.223 rows=222 loops=1)
                                                      Hash Cond: (cust_order.customerid =
customer_1.customerid)
                                                      ->  Seq Scan on cust_order  (cost=0.00..4.22 rows=222
width=12) (actual time=0.004..0.275 rows=222 loops=1)
                                                      ->  Hash  (cost=3.18..3.18 rows=118 width=4) (actual
time=0.326..0.326 rows=118 loops=1)
                                                            Buckets: 1024  Batches: 1  Memory Usage: 5kB
                                                            ->  Seq Scan on customer customer_1
(cost=0.00..3.18 rows=118 width=4) (actual time=0.003..0.153 rows=118 loops=1)
                                                ->  Hash  (cost=3.24..3.24 rows=124 width=8) (actual
time=0.351..0.351 rows=124 loops=1)
                                                      Buckets: 1024  Batches: 1  Memory Usage: 5kB
                                                      ->  Seq Scan on "time"  (cost=0.00..3.24 rows=124
width=8) (actual time=0.006..0.173 rows=124 loops=1)
                              ->  Hash  (cost=1.12..1.12 rows=12 width=9) (actual time=0.042..0.042
rows=12 loops=1)
                                    Buckets: 1024  Batches: 1  Memory Usage: 1kB
                                    ->  Seq Scan on book  (cost=0.00..1.12 rows=12 width=9) (actual
time=0.003..0.021 rows=12 loops=1)
                        ->  Hash  (cost=3.18..3.18 rows=118 width=46) (actual time=0.354..0.354
rows=118 loops=1)
                              Buckets: 1024  Batches: 1  Memory Usage: 9kB
                              ->  Seq Scan on customer  (cost=0.00..3.18 rows=118 width=46) (actual
time=0.007..0.171 rows=118 loops=1)
 Planning time: 1.608 ms
 Execution time: 24.567 ms
```

```
(39 rows)

wusong3=> VACUUM (ANALYZE) ;
WARNING:  skipping "pg_authid" --- only superuser can vacuum it
WARNING:  skipping "pg_database" --- only superuser can vacuum it
WARNING:  skipping "pg_db_role_setting" --- only superuser can vacuum it
WARNING:  skipping "pg_tablespace" --- only superuser can vacuum it
WARNING:  skipping "pg_pltemplate" --- only superuser can vacuum it
WARNING:  skipping "pg_auth_members" --- only superuser can vacuum it
WARNING:  skipping "pg_shdepend" --- only superuser can vacuum it
WARNING:  skipping "pg_shdescription" --- only superuser can vacuum it
WARNING:  skipping "pg_shseclabel" --- only superuser can vacuum it

VACUUM


--2. The Data Mart,
EXPLAIN ANALYZE
SELECT SUM (AMNT) AS money, customer.customerid , customer.l_name  , customer.f_name
from sales  NATURAL join customer
group by customer.customerid , customer.l_name  , customer.f_name
 ORDER BY money DESC limit 5;
VACUUM (ANALYZE) ;

wusong3=> EXPLAIN ANALYZE
wusong3-> SELECT SUM (AMNT) AS money, customer.customerid , customer.l_name  ,
customer.f_name   from sales  NATURAL join customer
wusong3-> group by customer.customerid , customer.l_name  , customer.f_name
wusong3->  ORDER BY money DESC limit 5;
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
------
 Limit  (cost=51.20..51.21 rows=5 width=51) (actual time=7.827..7.842 rows=5 loops=1)
   ->  Sort  (cost=51.20..51.50 rows=118 width=51) (actual time=7.823..7.828 rows=5 loops=1)
         Sort Key: (sum(sales.amnt))
         Sort Method: top-N heapsort  Memory: 25kB
         ->  HashAggregate  (cost=47.77..49.24 rows=118 width=51) (actual time=7.451..7.611
rows=104 loops=1)
               Group Key: customer.customerid, customer.l_name, customer.f_name
               ->  Hash Join  (cost=4.65..37.07 rows=1070 width=51) (actual time=0.381..4.743
rows=1070 loops=1)
                     Hash Cond: (sales.customerid = customer.customerid)
                     ->  Seq Scan on sales  (cost=0.00..17.70 rows=1070 width=9) (actual
time=0.008..1.361 rows=1070 loops=1)
```

```
                ->  Hash  (cost=3.18..3.18 rows=118 width=46) (actual time=0.355..0.355
rows=118 loops=1)
                      Buckets: 1024  Batches: 1  Memory Usage: 9kB
                      ->  Seq Scan on customer  (cost=0.00..3.18 rows=118 width=46) (actual
time=0.004..0.162 rows=118 loops=1)
 Planning time: 0.420 ms
 Execution time: 7.948 ms
(14 rows)

wusong3=> VACUUM (ANALYZE) ;
WARNING:  skipping "pg_authid" --- only superuser can vacuum it
WARNING:  skipping "pg_database" --- only superuser can vacuum it
WARNING:  skipping "pg_db_role_setting" --- only superuser can vacuum it
WARNING:  skipping "pg_tablespace" --- only superuser can vacuum it
WARNING:  skipping "pg_pltemplate" --- only superuser can vacuum it
WARNING:  skipping "pg_auth_members" --- only superuser can vacuum it
WARNING:  skipping "pg_shdepend" --- only superuser can vacuum it
WARNING:  skipping "pg_shdescription" --- only superuser can vacuum it
WARNING:  skipping "pg_shseclabel" --- only superuser can vacuum it
VACUUM


--3. The view View1,


EXPLAIN ANALYZE
SELECT SUM (AMNT) AS money FROM View1 GROUP BY customerId
 ORDER BY money DESC limit 5;
VACUUM ANALYZE ;

wusong3=> EXPLAIN ANALYZE
wusong3-> SELECT SUM (AMNT) AS money FROM View1 GROUP BY customerId
wusong3->  ORDER BY money DESC limit 5;
                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Limit  (cost=36.08..36.09 rows=5 width=9) (actual time=3.829..3.845 rows=5 loops=1)
   ->  Sort  (cost=36.08..36.34 rows=104 width=9) (actual time=3.824..3.828 rows=5 loops=1)
         Sort Key: (sum(amnt))
         Sort Method: top-N heapsort  Memory: 25kB
         ->  HashAggregate  (cost=33.05..34.35 rows=104 width=9) (actual time=3.452..3.622
rows=104 loops=1)
               Group Key: customerid
               ->  Seq Scan on view1  (cost=0.00..27.70 rows=1070 width=9) (actual
time=0.006..1.344 rows=1070 loops=1)
```

```
 Planning time: 0.213 ms
 Execution time: 3.924 ms
(9 rows)

wusong3=> VACUUM ANALYZE ;
WARNING:  skipping "pg_authid" --- only superuser can vacuum it
WARNING:  skipping "pg_database" --- only superuser can vacuum it
WARNING:  skipping "pg_db_role_setting" --- only superuser can vacuum it
WARNING:  skipping "pg_tablespace" --- only superuser can vacuum it
WARNING:  skipping "pg_pltemplate" --- only superuser can vacuum it
WARNING:  skipping "pg_auth_members" --- only superuser can vacuum it
WARNING:  skipping "pg_shdepend" --- only superuser can vacuum it
WARNING:  skipping "pg_shdescription" --- only superuser can vacuum it
WARNING:  skipping "pg_shseclabel" --- only superuser can vacuum it
VACUUM



--4. The view View2.

EXPLAIN ANALYZE
SELECT sum(sum) AS money ,  customerid , f_name , l_name FROM View2
group by customerid , f_name , l_name
 ORDER BY money DESC limit 5 ;
 VACUUM ANALYZE ;

 wusong3=> EXPLAIN ANALYZE
wusong3-> SELECT sum(sum) AS money ,  customerid , f_name , l_name FROM View2
wusong3-> group by customerid , f_name , l_name
wusong3->  ORDER BY money DESC limit 5 ;
                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Limit  (cost=7.67..7.68 rows=5 width=51) (actual time=0.928..0.944 rows=5 loops=1)
   ->  Sort  (cost=7.67..7.93 rows=104 width=51) (actual time=0.925..0.930 rows=5 loops=1)
         Sort Key: (sum(sum))
         Sort Method: top-N heapsort  Memory: 25kB
         ->  HashAggregate  (cost=4.64..5.94 rows=104 width=51) (actual time=0.551..0.703
rows=104 loops=1)
               Group Key: customerid, f_name, l_name
               ->  Seq Scan on view2  (cost=0.00..3.32 rows=132 width=51) (actual
time=0.011..0.193 rows=132 loops=1)
 Planning time: 0.144 ms
 Execution time: 1.025 ms
(9 rows)
```

```
wusong3=>  VACUUM ANALYZE ;
WARNING:  skipping "pg_authid" --- only superuser can vacuum it
WARNING:  skipping "pg_database" --- only superuser can vacuum it
WARNING:  skipping "pg_db_role_setting" --- only superuser can vacuum it
WARNING:  skipping "pg_tablespace" --- only superuser can vacuum it
WARNING:  skipping "pg_pltemplate" --- only superuser can vacuum it
WARNING:  skipping "pg_auth_members" --- only superuser can vacuum it
WARNING:  skipping "pg_shdepend" --- only superuser can vacuum it
WARNING:  skipping "pg_shdescription" --- only superuser can vacuum it
WARNING:  skipping "pg_shseclabel" --- only superuser can vacuum it
VACUUM
```

Explain the findings: the cost of the execution of the QUERY drops as more aggregate function perform in MATERILIAZED view level .

b)

```
drop  materialized view if exists View3 CASCADE;
CREATE MATERIALIZED VIEW View3 AS
SELECT District, TimeId, DayOfWeek, ISBN, SUM(Amnt)
FROM Sales NATURAL JOIN Customer NATURAL JOIN Time
GROUP BY District, TimeId, DayOfWeek, ISBN;
```

```
--1. The “Book Orders Database”
EXPLAIN ANALYZE
SELECT  SUM(Amnt) , country FROM
 customer NATURAL JOIN
(select customer.CustomerId as  CustomerId ,TIME.TimeId as TimeId   , book.isbn AS  ISBN ,
SUM(order_detail.quantity * book.price)  AS Amnt
from book NATURAL join order_detail NATURAL JOIN cust_order NATURAL JOIN customer
NATURAL JOIN TIME
group by  customer.CustomerId  ,  TIME.TimeId  ,book.isbn
order by  customer.CustomerId , TIME.TimeId   ,  book.isbn  ) AS sales
GROUP BY country order BY SUM DESC limit 1;
 VACUUM ANALYZE ;

 wusong3=> EXPLAIN ANALYZE
 wusong3-> SELECT  SUM(Amnt) , country FROM
 wusong3->  customer NATURAL JOIN
```

```
 wusong3-> (select customer.CustomerId as  CustomerId ,TIME.TimeId as TimeId   , book.isbn
AS  ISBN ,  SUM(order_detail.quantity * book.price)  AS Amnt
 wusong3(> from book NATURAL join order_detail NATURAL JOIN cust_order NATURAL JOIN
customer NATURAL JOIN TIME
 wusong3(> group by  customer.CustomerId  ,  TIME.TimeId  ,book.isbn
 wusong3(> order by  customer.CustomerId , TIME.TimeId   ,  book.isbn  ) AS sales
 wusong3-> GROUP BY country order BY SUM DESC limit 1;
                                                                    QUERY PLAN
 ------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------
  Limit  (cost=191.27..191.27 rows=1 width=48) (actual time=23.613..23.615 rows=1 loops=1)
    ->  Sort  (cost=191.27..191.29 rows=7 width=48) (actual time=23.608..23.608 rows=1
loops=1)
          Sort Key: (sum((sum(((order_detail.quantity)::numeric * book.price)))))
          Sort Method: top-N heapsort  Memory: 25kB
          ->  HashAggregate  (cost=191.15..191.24 rows=7 width=48) (actual time=23.540..23.552
rows=7 loops=1)
                Group Key: customer.country
                ->  Hash Join  (cost=133.29..187.90 rows=649 width=48) (actual time=12.594..21.422
rows=1070 loops=1)
                      Hash Cond: (customer_1.customerid = customer.customerid)
                      ->  GroupAggregate  (cost=128.63..161.63 rows=1100 width=28) (actual
time=12.226..18.079 rows=1070 loops=1)
                            Group Key: customer_1.customerid, "time".timeid, book.isbn
                            ->  Sort  (cost=128.63..131.38 rows=1100 width=28) (actual
time=12.200..13.565 rows=1100 loops=1)
                                  Sort Key: customer_1.customerid, "time".timeid, book.isbn
                                  Sort Method: quicksort  Memory: 134kB
                                  ->  Hash Join  (cost=23.82..73.06 rows=1100 width=28) (actual
time=2.617..9.985 rows=1100 loops=1)
                                        Hash Cond: (order_detail.isbn = book.isbn)
                                        ->  Hash Join  (cost=22.55..56.67 rows=1100 width=14) (actual
time=2.563..7.010 rows=1100 loops=1)
                                              Hash Cond: (order_detail.orderid = cust_order.orderid)
                                              ->  Seq Scan on order_detail  (cost=0.00..19.00 rows=1100
width=10) (actual time=0.008..1.410 rows=1100 loops=1)
                                              ->  Hash  (cost=19.77..19.77 rows=222 width=12) (actual
time=2.538..2.538 rows=222 loops=1)
                                                    Buckets: 1024  Batches: 1  Memory Usage: 10kB
                                                    ->  Hash Join  (cost=9.45..19.77 rows=222 width=12) (actual
time=0.712..2.198 rows=222 loops=1)
                                                          Hash Cond: (cust_order.orderdate = "time".orderdate)
                                                          ->  Hash Join  (cost=4.65..11.93 rows=222 width=12)
(actual time=0.350..1.221 rows=222 loops=1)
```

```
                                        Hash Cond: (cust_order.customerid =
customer_1.customerid)
                                        ->  Seq Scan on cust_order  (cost=0.00..4.22 rows=222
width=12) (actual time=0.008..0.279 rows=222 loops=1)
                                        ->  Hash  (cost=3.18..3.18 rows=118 width=4) (actual
time=0.327..0.327 rows=118 loops=1)
                                              Buckets: 1024  Batches: 1  Memory Usage: 5kB
                                              ->  Seq Scan on customer customer_1
(cost=0.00..3.18 rows=118 width=4) (actual time=0.005..0.157 rows=118 loops=1)
                                    ->  Hash  (cost=3.24..3.24 rows=124 width=8) (actual
time=0.350..0.350 rows=124 loops=1)
                                        Buckets: 1024  Batches: 1  Memory Usage: 5kB
                                        ->  Seq Scan on "time"  (cost=0.00..3.24 rows=124
width=8) (actual time=0.009..0.170 rows=124 loops=1)
                              ->  Hash  (cost=1.12..1.12 rows=12 width=18) (actual
time=0.042..0.042 rows=12 loops=1)
                                    Buckets: 1024  Batches: 1  Memory Usage: 1kB
                                    ->  Seq Scan on book  (cost=0.00..1.12 rows=12 width=18) (actual
time=0.005..0.019 rows=12 loops=1)
                  ->  Hash  (cost=3.18..3.18 rows=118 width=20) (actual time=0.352..0.352
rows=118 loops=1)
                        Buckets: 1024  Batches: 1  Memory Usage: 6kB
                        ->  Seq Scan on customer  (cost=0.00..3.18 rows=118 width=20) (actual
time=0.009..0.174 rows=118 loops=1)
 Planning time: 1.400 ms
 Execution time: 23.793 ms
(39 rows)

wusong3=>  VACUUM ANALYZE ;
WARNING:  skipping "pg_authid" --- only superuser can vacuum it
WARNING:  skipping "pg_database" --- only superuser can vacuum it
WARNING:  skipping "pg_db_role_setting" --- only superuser can vacuum it
WARNING:  skipping "pg_tablespace" --- only superuser can vacuum it
WARNING:  skipping "pg_pltemplate" --- only superuser can vacuum it
WARNING:  skipping "pg_auth_members" --- only superuser can vacuum it
WARNING:  skipping "pg_shdepend" --- only superuser can vacuum it
WARNING:  skipping "pg_shdescription" --- only superuser can vacuum it
WARNING:  skipping "pg_shseclabel" --- only superuser can vacuum it
VACUUM


--2. The Data Mart,
EXPLAIN ANALYZE
SELECT  SUM(Amnt) , country FROM
```

```
 customer NATURAL JOIN
sales
GROUP BY country order BY SUM DESC limit 1;
 VACUUM ANALYZE ;

 wusong3=> EXPLAIN ANALYZE
wusong3-> SELECT  SUM(Amnt) , country FROM
wusong3->  customer NATURAL JOIN
wusong3-> sales
wusong3-> GROUP BY country order BY SUM DESC limit 1;
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------
------
 Limit  (cost=42.54..42.54 rows=1 width=21) (actual time=6.780..6.782 rows=1 loops=1)
   ->  Sort  (cost=42.54..42.56 rows=7 width=21) (actual time=6.776..6.776 rows=1 loops=1)
         Sort Key: (sum(sales.amnt))
         Sort Method: top-N heapsort  Memory: 25kB
         ->  HashAggregate  (cost=42.42..42.50 rows=7 width=21) (actual time=6.736..6.745
rows=7 loops=1)
               Group Key: customer.country
               ->  Hash Join  (cost=4.65..37.07 rows=1070 width=21) (actual time=0.376..4.667
rows=1070 loops=1)
                     Hash Cond: (sales.customerid = customer.customerid)
                     ->  Seq Scan on sales  (cost=0.00..17.70 rows=1070 width=9) (actual
time=0.008..1.327 rows=1070 loops=1)
                     ->  Hash  (cost=3.18..3.18 rows=118 width=20) (actual time=0.350..0.350
rows=118 loops=1)
                           Buckets: 1024  Batches: 1  Memory Usage: 6kB
                           ->  Seq Scan on customer  (cost=0.00..3.18 rows=118 width=20) (actual
time=0.004..0.169 rows=118 loops=1)
 Planning time: 0.473 ms
 Execution time: 6.871 ms
(14 rows)

wusong3=>  VACUUM ANALYZE ;
WARNING:  skipping "pg_authid" --- only superuser can vacuum it
WARNING:  skipping "pg_database" --- only superuser can vacuum it
WARNING:  skipping "pg_db_role_setting" --- only superuser can vacuum it
WARNING:  skipping "pg_tablespace" --- only superuser can vacuum it
WARNING:  skipping "pg_pltemplate" --- only superuser can vacuum it
WARNING:  skipping "pg_auth_members" --- only superuser can vacuum it
WARNING:  skipping "pg_shdepend" --- only superuser can vacuum it
WARNING:  skipping "pg_shdescription" --- only superuser can vacuum it
WARNING:  skipping "pg_shseclabel" --- only superuser can vacuum it
```

VACUUM

--3. The view View2,
EXPLAIN ANALYZE
SELECT SUM(sum) , country FROM
 view2 NATURAL JOIN
 customer
GROUP BY   country order BY SUM DESC limit 1;
 VACUUM ANALYZE ;

 wusong3=> EXPLAIN ANALYZE
 wusong3-> SELECT SUM(sum) , country FROM
 wusong3-> view2 NATURAL JOIN
 wusong3-> customer
 wusong3-> GROUP BY   country order BY SUM DESC limit 1;
                                                QUERY PLAN

  Limit (cost=10.09..10.09 rows=1 width=21) (actual time=1.450..1.451 rows=1 loops=1)
    -> Sort (cost=10.09..10.09 rows=1 width=21) (actual time=1.445..1.445 rows=1 loops=1)
          Sort Key: (sum(view2.sum))
          Sort Method: top-N heapsort Memory: 25kB
          -> HashAggregate (cost=10.06..10.08 rows=1 width=21) (actual time=1.406..1.418
rows=7 loops=1)
                Group Key: customer.country
                -> Hash Join (cost=5.25..10.06 rows=1 width=21) (actual time=0.432..1.106
rows=132 loops=1)
                      Hash Cond: ((view2.customerid = customer.customerid) AND (view2.f_name =
customer.f_name) AND (view2.l_name = customer.l
 _name))
                      -> Seq Scan on view2 (cost=0.00..3.32 rows=132 width=51) (actual
time=0.007..0.174 rows=132 loops=1)
                      -> Hash (cost=3.18..3.18 rows=118 width=62) (actual time=0.400..0.400
rows=118 loops=1)
                            Buckets: 1024 Batches: 1 Memory Usage: 11kB
                            -> Seq Scan on customer (cost=0.00..3.18 rows=118 width=62) (actual
time=0.005..0.167 rows=118 loops=1)
  Planning time: 0.929 ms
  Execution time: 1.547 ms
 (14 rows)

```
 wusong3=>  VACUUM ANALYZE ;
 WARNING:  skipping "pg_authid" --- only superuser can vacuum it
 WARNING:  skipping "pg_database" --- only superuser can vacuum it
 WARNING:  skipping "pg_db_role_setting" --- only superuser can vacuum it
 WARNING:  skipping "pg_tablespace" --- only superuser can vacuum it
 WARNING:  skipping "pg_pltemplate" --- only superuser can vacuum it
 WARNING:  skipping "pg_auth_members" --- only superuser can vacuum it
 WARNING:  skipping "pg_shdepend" --- only superuser can vacuum it
 WARNING:  skipping "pg_shdescription" --- only superuser can vacuum it
 WARNING:  skipping "pg_shseclabel" --- only superuser can vacuum it
 VACUUM


--4. The view View3.

EXPLAIN ANALYZE
SELECT  SUM(sum) , country FROM
 view3  NATURAL JOIN
 (select distinct district , country from customer) as cc
GROUP BY country  order BY SUM DESC limit 1;
 VACUUM ANALYZE ;

 wusong3=> EXPLAIN ANALYZE
wusong3-> SELECT  SUM(sum) , country FROM
wusong3->  view3  NATURAL JOIN
wusong3->  (select distinct district , country from customer) as cc
wusong3-> GROUP BY country  order BY SUM DESC limit 1;
                                                    QUERY PLAN

-------------------------------------------------------------------------------------------------------------------------
--------------
----
 Limit  (cost=43.54..43.55 rows=1 width=21) (actual time=6.830..6.832 rows=1 loops=1)
   ->  Sort  (cost=43.54..43.59 rows=17 width=21) (actual time=6.826..6.826 rows=1 loops=1)
         Sort Key: (sum(view3.sum))
         Sort Method: top-N heapsort  Memory: 25kB
         ->  HashAggregate  (cost=43.25..43.46 rows=17 width=21) (actual time=6.786..6.798
rows=7 loops=1)
               Group Key: cc.country
               ->  Hash Join  (cost=4.32..38.22 rows=1006 width=21) (actual time=0.535..4.849
rows=1006 loops=1)
                     Hash Cond: (view3.district = cc.district)
                     ->  Seq Scan on view3  (cost=0.00..20.06 rows=1006 width=21) (actual
time=0.011..1.261 rows=1006 loops=1)
```

```
                ->  Hash  (cost=4.11..4.11 rows=17 width=32) (actual time=0.505..0.505 rows=17
loops=1)
                      Buckets: 1024  Batches: 1  Memory Usage: 2kB
                      ->  Subquery Scan on cc  (cost=3.77..4.11 rows=17 width=32) (actual
time=0.413..0.475 rows=17 loops=1)
                            ->  HashAggregate  (cost=3.77..3.94 rows=17 width=32) (actual
time=0.409..0.432 rows=17 loops=1)
                                  Group Key: customer.district, customer.country
                                  ->  Seq Scan on customer  (cost=0.00..3.18 rows=118 width=32)
(actual time=0.007..0.177 rows=118 loops
=1)
 Planning time: 0.294 ms
 Execution time: 6.939 ms
(17 rows)

wusong3=>  VACUUM ANALYZE ;
WARNING:  skipping "pg_authid" --- only superuser can vacuum it
WARNING:  skipping "pg_database" --- only superuser can vacuum it
WARNING:  skipping "pg_db_role_setting" --- only superuser can vacuum it
WARNING:  skipping "pg_tablespace" --- only superuser can vacuum it
WARNING:  skipping "pg_pltemplate" --- only superuser can vacuum it
WARNING:  skipping "pg_auth_members" --- only superuser can vacuum it
WARNING:  skipping "pg_shdepend" --- only superuser can vacuum it
WARNING:  skipping "pg_shdescription" --- only superuser can vacuum it
WARNING:  skipping "pg_shseclabel" --- only superuser can vacuum it
VACUUM
```

Explain the findings: the cost of the execution of the QUERY drops as more aggregate function perform in MATERILIAZED view level .

Question 5 .

a)

```
SELECT DISTINCT * FROM (
 SELECT customerid ,  f_name ,   city , sum(amnt) OVER ws, avg(amnt) OVER wa
   FROM sales NATURAL join time NATURAL JOIN customer
    WHERE Month in ('April', 'May' )and Year = 2017
   WINDOW ws AS (PARTITION BY customerid ) ,
      wa AS (PARTITION BY city )
 )   AS T ORDER BY CITY  ;
```

B)

```
DROP MATERIALIZED VIEW IF EXISTS V1 CASCADE;
CREATE MATERIALIZED VIEW V1 AS
  SELECT  city,  timeid,  OrderDate AS day,    sum(amnt) AS ST
  FROM sales NATURAL JOIN customer NATURAL JOIN  time
  WHERE  Year = 2017 AND Month IN ('April', 'May')
  GROUP BY city, timeid, OrderDate
  ORDER BY city, timeid , OrderDate;

SELECT  city,  timeid,  day,  ST AS "sum(amnt)", sum(ST) OVER W1 AS cumulative_sum
FROM V1 WINDOW W1 AS ( PARTITION BY city ORDER BY timeid )
ORDER BY city, timeid , day ,st;
```