Γεώργιος Ρουγκάλας Π21144

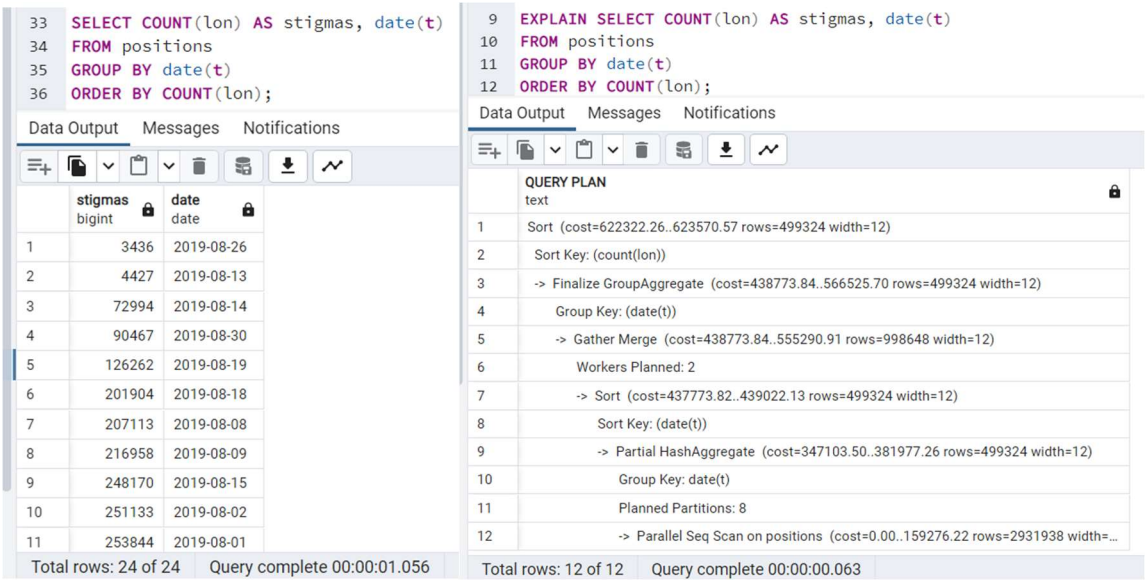
Ερώτημα 1

i. SELECT COUNT(lon) AS stigmas, date(t)

FROM positions

GROUP BY date(t)

ORDER BY COUNT(lon);



```
33 SELECT COUNT(lon) AS stigmas, date(t)
34 FROM positions
35 GROUP BY date(t)
36 ORDER BY COUNT(lon);
```

Data Output | Messages | Notifications

	stigmas bigint	date date
1	3436	2019-08-26
2	4427	2019-08-13
3	72994	2019-08-14
4	90467	2019-08-30
5	126262	2019-08-19
6	201904	2019-08-18
7	207113	2019-08-08
8	216958	2019-08-09
9	248170	2019-08-15
10	251133	2019-08-02
11	253844	2019-08-01

Total rows: 24 of 24 Query complete 00:00:01.056

```
9  EXPLAIN SELECT COUNT(lon) AS stigmas, date(t)
10 FROM positions
11 GROUP BY date(t)
12 ORDER BY COUNT(lon);
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Sort (cost=622322.26..623570.57 rows=499324 width=12)
2	Sort Key: (count(lon))
3	-> Finalize GroupAggregate (cost=438773.84..566525.70 rows=499324 width=12)
4	Group Key: (date(t))
5	-> Gather Merge (cost=438773.84..555290.91 rows=998648 width=12)
6	Workers Planned: 2
7	-> Sort (cost=437773.82..439022.13 rows=499324 width=12)
8	Sort Key: (date(t))
9	-> Partial HashAggregate (cost=347103.50..381977.26 rows=499324 width=12)
10	Group Key: date(t)
11	Planned Partitions: 8
12	-> Parallel Seq Scan on positions (cost=0.00..159276.22 rows=2931938 width=...

Total rows: 12 of 12 Query complete 00:00:00.063

ii. SELECT type, COUNT(id) AS greek_ships

FROM vessels

where flag = 'Greece'

GROUP BY type

ORDER BY type;

```
41 SELECT type, COUNT(id) AS greek_ships
42 FROM vessels
43 where flag = 'Greece'
44 GROUP BY type
45 ORDER BY type;
```

Data Output | Messages | Notifications

	type numeric (3)	greek_ships bigint
1	0	4
2	4	1
3	25	1
4	30	6
5	31	4
6	36	29
7	37	25
8	40	4
9	42	1
10	49	3
11	50	4

Total rows: 29 of 29 Query complete 00:00:00.084

```
14 EXPLAIN
15 SELECT type, COUNT(id) AS greek_ships
16 FROM vessels
17 where flag = 'Greece'
18 GROUP BY type
19 ORDER BY type;
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Sort (cost=15.49..15.58 rows=33 width=12)
2	Sort Key: type
3	-> HashAggregate (cost=14.33..14.66 rows=33 width=12)
4	Group Key: type
5	-> Seq Scan on vessels (cost=0.00..13.11 rows=244 width=6...
6	Filter: ((flag)::text = 'Greece'::text)

Total rows: 6 of 6 Query complete 00:00:00.072

iii. with shipcount(ship_type, total) as (

select type, count(distinct(vessel_id))

from positions, vessels, vesseltypes

where speed > 30 and vessel_id = vessels.id and type = code

group by type

)

select distinct(vessel_id), type, description, total as total_of_ships_by_type

from positions, vessels, vesseltypes, shipcount

where speed > 30 and vessel_id = vessels.id and type = code and code = ship_type;

```
27  with shipcount(ship_type, total) as (
28      select type, count(distinct(vessel_id))
29      from positions, vessels, vesseltypes
30      where speed > 30 and vessel_id = vessels.id and type = code
31      group by type
32      )
33  select distinct(vessel_id), type, description, total as total_of_ships_by_type
34  from positions, vessels, vesseltypes, shipcount
35  where speed > 30 and vessel_id = vessels.id and type = code and code = ship_type;
```

Data Output | Messages | Notifications

	vessel_id character varying	type numeric (3)	description character varying	total_of_ships_by_type bigint
1	1eee5599ef5de5c07df6...	40	High speed craft (HSC), all ships of this type	6
2	26199b57ef2b787a02bf...	80	Tanker, all ships of this type	2
3	3fc28f4d2b3c7cb5f68d...	49	High speed craft (HSC), No additional information	2
4	4babbcb0d3d48cf988de...	31	Towing	1
5	53880c54896ed4abe0fc...	49	High speed craft (HSC), No additional information	2
6	57139e85177dea04c5a...	42	High speed craft (HSC), Hazardous category B	1

Total rows: 16 of 16 | Query complete 00:00:02.162

```
26 explain
27 with shipcount(ship_type, total) as (
28     select type, count(distinct(vessel_id))
29     from positions, vessels, vesseltypes
30     where speed > 30 and vessel_id = vessels.id and type = code
31     group by type
32     )
33 select distinct(vessel_id), type, description, total as total_of_ships_by_type
34 from positions, vessels, vesseltypes, shipcount
35 where speed > 30 and vessel_id = vessels.id and type = code and code = ship_type;
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Unique (cost=409486.79..411961.27 rows=197958 width=107)
2	-> Sort (cost=409486.79..409981.69 rows=197958 width=107)
3	Sort Key: positions.vessel_id, vessels.type, vesseltypes.description, (count(DISTINCT positions_1.vessel_id))
4	-> Hash Join (cost=196943.42..380567.10 rows=197958 width=107)
5	Hash Cond: ((positions.vessel_id)::text = (vessels.id)::text)
6	-> Gather (cost=1000.00..181835.62 rows=215594 width=65)
7	Workers Planned: 2
8	-> Parallel Seq Scan on positions (cost=0.00..159276.22 rows=89831 width=65)
9	Filter: (speed > '30'::numeric)
10	-> Hash (cost=195937.81..195937.81 rows=449 width=107)
11	-> Merge Join (cost=171895.62..195937.81 rows=449 width=107)
12	Merge Cond: (vessels.type = vesseltypes.code)
13	-> Merge Join (cost=171890.00..198085.12 rows=449 width=81)
14	Merge Cond: (vessels.type = vessels_1.type)
15	-> Sort (cost=33.73..34.96 rows=489 width=69)
16	Sort Key: vessels.type
17	-> Seq Scan on vessels (cost=0.00..11.89 rows=489 width=69)
18	-> GroupAggregate (cost=171856.26..198044.04 rows=33 width=12)
19	Group Key: vessels_1.type
20	-> Gather Merge (cost=171856.26..196965.74 rows=215594 width=69)
21	Workers Planned: 2
22	-> Sort (cost=170856.24..171080.82 rows=89831 width=69)
23	Sort Key: vessels_1.type
24	-> Hash Join (cost=21.39..159779.93 rows=89831 width=69)
25	Hash Cond: (vessels_1.type = vesseltypes_1.code)
26	-> Hash Join (cost=18.00..159532.08 rows=89831 width=69)
27	Hash Cond: ((positions_1.vessel_id)::text = (vessels_1.id)::text)
28	-> Parallel Seq Scan on positions positions_1 (cost=0.00..159276.22 rows=89831 width…
29	Filter: (speed > '30'::numeric)
30	-> Hash (cost=11.89..11.89 rows=489 width=69)
31	-> Seq Scan on vessels vessels_1 (cost=0.00..11.89 rows=489 width=69)
32	-> Hash (cost=2.06..2.06 rows=106 width=4)
33	-> Seq Scan on vesseltypes vesseltypes_1 (cost=0.00..2.06 rows=106 width=4)
34	-> Sort (cost=5.63..5.89 rows=106 width=34)
35	Sort Key: vesseltypes.code
36	-> Seq Scan on vesseltypes (cost=0.00..2.06 rows=106 width=34)

Total rows: 36 of 36 | Query complete 00:00:00.052

iv. SELECT date(t), COUNT(lon) AS stigmas

FROM positions, vessels, vesseltypes

WHERE (date(t) BETWEEN '2019-08-14' and '2019-08-18')

AND (vessel_id = vessels.id AND type = code AND description LIKE 'Passenger%')

GROUP BY date(t)

ORDER BY date(t);

```
49  SELECT date(t), COUNT(lon) AS stigmas
50  FROM positions, vessels, vesseltypes
51  WHERE (date(t) BETWEEN '2019-08-14' and '2019-08-18')
52  AND (vessel_id = vessels.id AND type = code AND description LIKE 'Passenger%')
53  GROUP BY date(t)
54  ORDER BY date(t);
```

Data Output | Messages | Notifications

	date date	stigmas bigint
1	2019-08-14	20933
2	2019-08-15	69960
3	2019-08-16	109647
4	2019-08-17	97092
5	2019-08-18	63861

Total rows: 5 of 5 | Query complete 00:00:00.764

```
48  EXPLAIN
49  SELECT date(t), COUNT(lon) AS stigmas
50  FROM positions, vessels, vesseltypes
51  WHERE (date(t) BETWEEN '2019-08-14' and '2019-08-18')
52  AND (vessel_id = vessels.id AND type = code AND description LIKE 'Passenger%')
53  GROUP BY date(t)
54  ORDER BY date(t);
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Finalize GroupAggregate (cost=182426.40..182828.64 rows=3319 width=12)
2	Group Key: (date(positions.t))
3	-> Gather Merge (cost=182426.40..182773.32 rows=2766 width=12)
4	Workers Planned: 2
5	-> Partial GroupAggregate (cost=181426.37..181454.03 rows=1383 width=12)
6	Group Key: (date(positions.t))
7	-> Sort (cost=181426.37..181429.83 rows=1383 width=12)
8	Sort Key: (date(positions.t))
9	-> Hash Join (cost=16.24..181354.22 rows=1383 width=12)
10	Hash Cond: ((positions.vessel_id)::text = (vessels.id)::text)
11	-> Parallel Seq Scan on positions (cost=0.00..181265.76 rows=14660 width=81)
12	Filter: ((date(t) >= '2019-08-14'::date) AND (date(t) <= '2019-08-18'::date))
13	-> Hash (cost=15.67..15.67 rows=46 width=65)
14	-> Hash Join (cost=2.45..15.67 rows=46 width=65)
15	Hash Cond: (vessels.type = vesseltypes.code)
16	-> Seq Scan on vessels (cost=0.00..11.89 rows=489 width=69)
17	-> Hash (cost=2.33..2.33 rows=10 width=4)
18	-> Seq Scan on vesseltypes (cost=0.00..2.33 rows=10 width=4)
19	Filter: ((description)::text ~~ 'Passenger%'::text)

Total rows: 19 of 19 | Query complete 00:00:00.081

v. /* Πιστεύω ότι θα ήταν καλύτερο αν τα δύο υπο-ερωτήματα γραφόντουσαν σε δύο διαφορετικές αναζητήσεις, παρόλα αυτά τα συνδύασα όπως ζητήθηκε από την εκφώνηση */

```
WITH question1(answer1) AS (
        SELECT Distinct(vessel_id)
        FROM positions, vessels, vesseltypes
        WHERE (date(t) BETWEEN '2019-08-15' and '2019-08-18')
        AND (vessel_id = vessels.id AND type = code AND description LIKE 'Cargo%')
        ),
totalspeeds(ships, distance) AS (
        SELECT vessel_id, SUM(speed)
        FROM positions
        WHERE (date(t) BETWEEN '2019-08-12' AND '2019-08-19')
        GROUP BY vessel_id
        ),
question2(answer2) AS (
        SELECT DISTINCT(vessel_id)
        FROM positions, vessels, vesseltypes, totalspeeds
        WHERE vessel_id = ships AND distance = 0
        AND (vessel_id = vessels.id AND type = code AND description LIKE 'Cargo%')
        )
SELECT answer1 as stopped_once, answer2 as remained_stoped
FROM question1, question2;
```

```
78  WITH question1(answer1) AS (
79      SELECT Distinct(vessel_id)
80      FROM positions, vessels, vesseltypes
81      WHERE (date(t) BETWEEN '2019-08-15' and '2019-08-18')
82      AND (vessel_id = vessels.id AND type = code AND description LIKE 'Cargo%')
83      ),
84  totalspeeds(ships, distance) AS (
85      SELECT vessel_id, SUM(speed)
86      FROM positions
87      WHERE (date(t) BETWEEN '2019-08-12' AND '2019-08-19')
88      GROUP BY vessel_id
89      ),
90  question2(answer2) AS (
91      SELECT DISTINCT(vessel_id)
92      FROM positions, vessels, vesseltypes, totalspeeds
93      WHERE vessel_id = ships AND distance = 0
94      AND (vessel_id = vessels.id AND type = code AND description LIKE 'Cargo%')
95      )
96  SELECT answer1 as stopped_once, answer2 as remained_stoped
97  FROM question1, question2;
```

Data Output | Messages | Notifications

	stopped_once character varying	remained_stoped character varying
1	01c8d531d4f3dbf4d2ab71f0018d47aa0db2358fb5...	9a07029a6294dcf984fba483879732a7b9cc864ef7cd7000...
2	04b6c84a518f833b7206a114ada7660d56888de1a...	9a07029a6294dcf984fba483879732a7b9cc864ef7cd7000...
3	05edaaf038e80b4952faf8ea6ee12f2e95066896dc...	9a07029a6294dcf984fba483879732a7b9cc864ef7cd7000...
4	065809b408897ba8c70edf8ebd41aa8a52517815c...	9a07029a6294dcf984fba483879732a7b9cc864ef7cd7000...

Total rows: 39 of 39 Query complete 00:00:06.066

```
78  EXPLAIN
79  WITH question1(answer1) AS (
80      SELECT Distinct(vessel_id)
81      FROM positions, vessels, vesseltypes
82      WHERE (date(t) BETWEEN '2019-08-15' and '2019-08-18')
83      AND (vessel_id = vessels.id AND type = code AND description LIKE 'Cargo%')
84      ),
85  totalspeeds(ships, distance) AS (
86      SELECT vessel_id, SUM(speed)
87      FROM positions
88      WHERE (date(t) BETWEEN '2019-08-12' AND '2019-08-19')
89      GROUP BY vessel_id
90      ),
91  question2(answer2) AS (
92      SELECT DISTINCT(vessel_id)
93      FROM positions, vessels, vesseltypes, totalspeeds
94      WHERE vessel_id = ships AND distance = 0
95      AND (vessel_id = vessels.id AND type = code AND description LIKE 'Cargo%')
96      )
97  SELECT answer1 as stopped_once, answer2 as remained_stoped
98  FROM question1, question2;
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Nested Loop (cost=578805.38..581286.05 rows=180625 width=130)
2	-> Unique (cost=183297.16..183494.84 rows=425 width=65)
3	-> Gather Merge (cost=183297.16..183492.72 rows=850 width=65)

4	Workers Planned: 2
5	-> Unique (cost=182297.14..182394.58 rows=425 width=65)
6	-> Merge Join (cost=182297.14..182391.12 rows=1383 width=65)
7	Merge Cond: ((positions.vessel_id)::text = (vessels.id)::text)
8	-> Sort (cost=182280.20..182316.85 rows=14660 width=65)
9	Sort Key: positions.vessel_id
10	-> Parallel Seq Scan on positions (cost=0.00..181265.76 rows=14660 width=65)
11	Filter: ((date(t) >= '2019-08-15'::date) AND (date(t) <= '2019-08-18'::date))
12	-> Sort (cost=16.94..17.05 rows=46 width=65)
13	Sort Key: vessels.id
14	-> Hash Join (cost=2.45..15.67 rows=46 width=65)
15	Hash Cond: (vessels.type = vesseltypes.code)
16	-> Seq Scan on vessels (cost=0.00..11.89 rows=489 width=69)
17	-> Hash (cost=2.33..2.33 rows=10 width=4)
18	-> Seq Scan on vesseltypes (cost=0.00..2.33 rows=10 width=4)
19	Filter: ((description)::text ~~ 'Cargo%'::text)
20	-> Materialize (cost=395508.21..395530.21 rows=425 width=65)
21	-> Unique (cost=395508.21..395523.83 rows=425 width=65)
22	-> Sort (cost=395508.21..395516.02 rows=3124 width=65)
23	Sort Key: positions_1.vessel_id
24	-> Hash Join (cost=183522.64..395326.88 rows=3124 width=65)
25	Hash Cond: ((positions_1.vessel_id)::text = (vessels_1.id)::text)
26	-> Hash Join (cost=183506.39..395155.31 rows=33114 width=130)
27	Hash Cond: ((positions_1.vessel_id)::text = (totalspeeds.ships)::text)
28	-> Seq Scan on positions positions_1 (cost=0.00..192993.51 rows=7036651 width=65)
29	-> Hash (cost=183506.37..183506.37 rows=2 width=65)
30	-> Subquery Scan on totalspeeds (cost=183280.22..183506.37 rows=2 width=65)
31	-> Finalize GroupAggregate (cost=183280.22..183506.35 rows=2 width=97)
32	Group Key: positions_2.vessel_id
33	Filter: (sum(positions_2.speed) = '0'::numeric)
34	-> Gather Merge (cost=183280.22..183493.60 rows=850 width=97)
35	Workers Planned: 2
36	-> Partial GroupAggregate (cost=182280.20..182395.46 rows=425 width=97)
37	Group Key: positions_2.vessel_id
38	-> Sort (cost=182280.20..182316.85 rows=14660 width=69)
39	Sort Key: positions_2.vessel_id
40	-> Parallel Seq Scan on positions positions_2 (cost=0.00..181265.76 rows=14660 width=69)
41	Filter: ((date(t) >= '2019-08-12'::date) AND (date(t) <= '2019-08-19'::date))
42	-> Hash (cost=15.67..15.67 rows=46 width=65)
43	-> Hash Join (cost=2.45..15.67 rows=46 width=65)
44	Hash Cond: (vessels_1.type = vesseltypes_1.code)
45	-> Seq Scan on vessels vessels_1 (cost=0.00..11.89 rows=489 width=69)
46	-> Hash (cost=2.33..2.33 rows=10 width=4)
47	-> Seq Scan on vesseltypes vesseltypes_1 (cost=0.00..2.33 rows=10 width=4)
48	Filter: ((description)::text ~~ 'Cargo%'::text)

Total rows: 48 of 48 Query complete 00:00:00.053

Ερώτημα 2

i.

```
14  ALTER SYSTEM SET shared_buffers TO '256MB';
15  SELECT COUNT(lon) AS stigmas, date(t)
16  FROM positions
17  GROUP BY date(t)
18  ORDER BY COUNT(lon);
```

Data Output | Messages | Notifications

	stigmas bigint	date date
1	3436	2019-08-26
2	4427	2019-08-13
3	72994	2019-08-14
4	90467	2019-08-30
5	126262	2019-08-19
6	201904	2019-08-18

Total rows: 24 of 24 | Query complete 00:00:00.862

```
14  ALTER SYSTEM SET shared_buffers TO '256MB';
15  EXPLAIN
16  SELECT COUNT(lon) AS stigmas, date(t)
17  FROM positions
18  GROUP BY date(t)
19  ORDER BY COUNT(lon);
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Sort (cost=622322.26..623570.57 rows=499324 width=12)
2	Sort Key: (count(lon))
3	-> Finalize GroupAggregate (cost=438773.84..566525.70 rows=499324 width=12)
4	Group Key: (date(t))
5	-> Gather Merge (cost=438773.84..555290.91 rows=998648 width=12)
6	Workers Planned: 2

Total rows: 12 of 12 | Query complete 00:00:00.046

ii.

```
13  ALTER SYSTEM SET shared_buffers TO '256MB';
14  SELECT type, COUNT(id) AS greek_ships
15  FROM vessels
16  where flag = 'Greece'
17  GROUP BY type
18  ORDER BY type;
```

Data Output | Messages | Notifications

	type numeric (3)	greek_ships bigint
1	0	4
2	4	1
3	25	1
4	30	6
5	31	4
6	36	29

Total rows: 29 of 29 | Query complete 00:00:00.061

```
13  ALTER SYSTEM SET shared_buffers TO '256MB';
14  EXPLAIN
15  SELECT type, COUNT(id) AS greek_ships
16  FROM vessels
17  where flag = 'Greece'
18  GROUP BY type
19  ORDER BY type;
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Sort (cost=15.49..15.58 rows=33 width=12)
2	Sort Key: type
3	-> HashAggregate (cost=14.33..14.66 rows=33 width=12)
4	Group Key: type
5	-> Seq Scan on vessels (cost=0.00..13.11 rows=244 width=6...
6	Filter: ((flag)::text = 'Greece'::text)

Total rows: 6 of 6 | Query complete 00:00:00.042

iii.

```
13  ALTER SYSTEM SET shared_buffers TO '256MB';
14  with shipcount(ship_type, total) as (
15      select type, count(distinct(vessel_id))
16      from positions, vessels, vesseltypes
17      where speed > 30 and vessel_id = vessels.id and type = code
18      group by type
19      )
20  select distinct(vessel_id), type, description, total as total_of_ships_by_type
21  from positions, vessels, vesseltypes, shipcount
22  where speed > 30 and vessel_id = vessels.id and type = code and code = ship_type;
```

Data Output | Messages | Notifications

	vessel_id character varying	type numeric (3)	description character varying	total_of_ships_by_type bigint
1	1eee5599ef5de5c07df6...	40	High speed craft (HSC), all ships of this type	6
2	26199b57ef2b787a02bf...	80	Tanker, all ships of this type	2
3	3fc28f4d2b3c7cb5f68d5...	49	High speed craft (HSC), No additional information	2
4	4babbcb0d3d48cf988de...	31	Towing	1
5	53880c54896ed4abe0fc...	49	High speed craft (HSC), No additional information	2
6	57139e85177dea04c5a3...	42	High speed craft (HSC), Hazardous category B	1

Total rows: 16 of 16 Query complete 00:00:01.893

```
12  ALTER SYSTEM SET shared_buffers TO '256MB';
13  EXPLAIN
14  with shipcount(ship_type, total) as (
15      select type, count(distinct(vessel_id))
16      from positions, vessels, vesseltypes
17      where speed > 30 and vessel_id = vessels.id and type = code
18      group by type
19      )
20  select distinct(vessel_id), type, description, total as total_of_ships_by_type
21  from positions, vessels, vesseltypes, shipcount
22  where speed > 30 and vessel_id = vessels.id and type = code and code = ship_type;
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Unique (cost=409486.79..411961.27 rows=197958 width=107)
2	-> Sort (cost=409486.79..409981.69 rows=197958 width=107)
3	Sort Key: positions.vessel_id, vessels.type, vesseltypes.description, (count(DISTINCT positions_1.vessel_id))
4	-> Hash Join (cost=196943.42..380567.10 rows=197958 width=107)
5	Hash Cond: ((positions.vessel_id)::text = (vessels.id)::text)
6	-> Gather (cost=1000.00..181835.62 rows=215594 width=65)

Total rows: 36 of 36 Query complete 00:00:00.047

iv.

```
14  SELECT date(t), COUNT(lon) AS stigmas
15  FROM positions, vessels, vesseltypes
16  WHERE (date(t) BETWEEN '2019-08-14' and '2019-08-18')
17  AND (vessel_id = vessels.id AND type = code AND description LIKE 'Passenger%')
18  GROUP BY date(t)
19  ORDER BY date(t);
```

Data Output | Messages | Notifications

	date date	stigmas bigint
1	2019-08-14	20933
2	2019-08-15	69960
3	2019-08-16	109647
4	2019-08-17	97092
5	2019-08-18	63861

Total rows: 5 of 5 | Query complete 00:00:00.700

```
13  EXPLAIN
14  SELECT date(t), COUNT(lon) AS stigmas
15  FROM positions, vessels, vesseltypes
16  WHERE (date(t) BETWEEN '2019-08-14' and '2019-08-18')
17  AND (vessel_id = vessels.id AND type = code AND description LIKE 'Passenger%')
18  GROUP BY date(t)
19  ORDER BY date(t);
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Finalize GroupAggregate (cost=182426.40..182828.64 rows=3319 width=12)
2	Group Key: (date(positions.t))
3	-> Gather Merge (cost=182426.40..182773.32 rows=2766 width=12)
4	Workers Planned: 2
5	-> Partial GroupAggregate (cost=181426.37..181454.03 rows=1383 width=12)

Total rows: 19 of 19 | Query complete 00:00:00.050

v.

```
13  ALTER SYSTEM SET shared_buffers TO '256MB';
14  WITH question1(answer1) AS (
15      SELECT Distinct(vessel_id)
16      FROM positions, vessels, vesseltypes
17      WHERE (date(t) BETWEEN '2019-08-15' and '2019-08-18')
18      AND (vessel_id = vessels.id AND type = code AND description LIKE 'Cargo%')
19      ),
20  totalspeeds(ships, distance) AS (
21      SELECT vessel_id, SUM(speed)
22      FROM positions
23      WHERE (date(t) BETWEEN '2019-08-12' AND '2019-08-19')
24      GROUP BY vessel_id
25      ),
26  question2(answer2) AS (
27      SELECT DISTINCT(vessel_id)
28      FROM positions, vessels, vesseltypes, totalspeeds
29      WHERE vessel_id = ships AND distance = 0
30      AND (vessel_id = vessels.id AND type = code AND description LIKE 'Cargo%')
31      )
32  SELECT answer1 as stopped_once, answer2 as remained_stoped
33  FROM question1, question2;
```

Data Output | Messages | Notifications

	stopped_once character varying	remained_stoped character varying
1	01c8d531d4f3dbf4d2ab71f0018d47aa0db235…	9a07029a6294dcf984fba483879732a7b9cc864ef…
2	04b6c84a518f833b7206a114ada7660d56888d…	9a07029a6294dcf984fba483879732a7b9cc864ef…
3	05edaaf038e80b4952faf8ea6ee12f2e9506689…	9a07029a6294dcf984fba483879732a7b9cc864ef…

Total rows: 39 of 39 Query complete 00:00:05.264

```
12  ALTER SYSTEM SET shared_buffers TO '256MB';
13  EXPLAIN
14  WITH question1(answer1) AS (
15      SELECT Distinct(vessel_id)
16      FROM positions, vessels, vesseltypes
17      WHERE (date(t) BETWEEN '2019-08-15' and '2019-08-18')
18      AND (vessel_id = vessels.id AND type = code AND description LIKE 'Cargo%')
19      ),
20  totalspeeds(ships, distance) AS (
21      SELECT vessel_id, SUM(speed)
22      FROM positions
23      WHERE (date(t) BETWEEN '2019-08-12' AND '2019-08-19')
24      GROUP BY vessel_id
25      ),
26  question2(answer2) AS (
27      SELECT DISTINCT(vessel_id)
28      FROM positions, vessels, vesseltypes, totalspeeds
29      WHERE vessel_id = ships AND distance = 0
30      AND (vessel_id = vessels.id AND type = code AND description LIKE 'Cargo%')
31      )
32  SELECT answer1 as stopped_once, answer2 as remained_stoped
33  FROM question1, question2;
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Nested Loop (cost=578805.38..581286.05 rows=180625 width=130)
2	-> Unique (cost=183297.16..183494.84 rows=425 width=65)
3	-> Gather Merge (cost=183297.16..183492.72 rows=850 width=65)

Total rows: 48 of 48 Query complete 00:00:00.047

Παρατηρώ ότι οι χρόνοι βελτιώθηκαν σε όλα τα ερωτήματα. Επίσης, ο χρόνος που κέρδισε το πρόγραμμα για κάθε εκτέλεση μεγαλώνει αναλόγως με τον χρόνο που χρειάστηκε για την ίδια εκτέλεση στο πρώτο ερώτημα. Άρα, η μείωση του χρόνου είναι ποσοστιαία και όχι ένα σταθερό διάστημα χρόνου. Από αυτά, συμπεραίνω ότι υπήρξε αύξηση της ταχύτητας εκτέλεσης.

Ερώτημα 3

i.

```
6  SET max_parallel_workers_per_gather = 1024;
7  EXPLAIN
8  SELECT COUNT(lon) AS stigmas, date(t)
9  FROM positions
10 GROUP BY date(t)
11 ORDER BY COUNT(lon);
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Sort (cost=683581.86..684830.17 rows=499324 width=12)
2	Sort Key: (count(lon))
3	-> Finalize GroupAggregate (cost=307157.35..627785.30 rows=499324 width=12)
4	Group Key: (date(t))
5	-> Gather Merge (cost=307157.35..609060.65 rows=2496620 width=12)
6	Workers Planned: 5
7	-> Sort (cost=306157.27..307405.58 rows=499324 width=12)
8	Sort Key: (date(t))
9	-> Partial HashAggregate (cost=230375.71..250360.71 rows=499324 width=12)
10	Group Key: date(t)
11	Planned Partitions: 8
12	-> Parallel Seq Scan on positions (cost=0.00..140218.63 rows=1407330 width=12)

Total rows: 12 of 12 | Query complete 00:00:00.049

ii.

```
12 SET max_parallel_workers_per_gather = 1024;
13 EXPLAIN
14 SELECT type, COUNT(id) AS greek_ships
15 FROM vessels
16 where flag = 'Greece'
17 GROUP BY type
18 ORDER BY type;
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Sort (cost=15.49..15.58 rows=33 width=12)
2	Sort Key: type
3	-> HashAggregate (cost=14.33..14.66 rows=33 width=12)
4	Group Key: type
5	-> Seq Scan on vessels (cost=0.00..13.11 rows=244 width=69)
6	Filter: ((flag)::text = 'Greece'::text)

Total rows: 6 of 6 | Query complete 00:00:00.125

iii.

```
12  SET max_parallel_workers_per_gather = 1024;
13  EXPLAIN
14  with shipcount(ship_type, total) as (
15      select type, count(distinct(vessel_id))
16      from positions, vessels, vesseltypes
17      where speed > 30 and vessel_id = vessels.id and type = code
18      group by type
19      )
20  select distinct(vessel_id), type, description, total as total_of_ships_by_type
21  from positions, vessels, vesseltypes, shipcount
22  where speed > 30 and vessel_id = vessels.id and type = code and code = ship_type;
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Unique (cost=364245.83..366720.30 rows=197958 width=107)
2	-> Sort (cost=364245.83..364740.72 rows=197958 width=107)
3	Sort Key: positions.vessel_id, vessels.type, vesseltypes.description, (count(DISTINCT positions_1.vessel_id))
4	-> Hash Join (cost=170760.05..335326.14 rows=197958 width=107)
5	Hash Cond: ((positions.vessel_id)::text = (vessels.id)::text)
6	-> Gather (cost=1000.00..162778.03 rows=215594 width=65)
7	Workers Planned: 5
8	-> Parallel Seq Scan on positions (cost=0.00..140218.63 rows=43119 width=65)
9	Filter: (speed > '30'::numeric)
10	-> Hash (cost=169754.44..169754.44 rows=449 width=107)
11	-> Merge Join (cost=144830.27..169754.44 rows=449 width=107)
12	Merge Cond: (vessels.type = vesseltypes.code)
13	-> Merge Join (cost=144824.65..171980.96 rows=449 width=81)
14	Merge Cond: (vessels.type = vessels_1.type)
15	-> Sort (cost=33.73..34.96 rows=489 width=69)
16	Sort Key: vessels.type
17	-> Seq Scan on vessels (cost=0.00..11.89 rows=489 width=69)
18	-> GroupAggregate (cost=144790.92..171939.88 rows=33 width=12)
19	Group Key: vessels_1.type
20	-> Gather Merge (cost=144790.92..170861.58 rows=215594 width=69)
21	Workers Planned: 5
22	-> Sort (cost=143790.84..143898.64 rows=43119 width=69)
23	Sort Key: vessels_1.type
24	-> Hash Join (cost=21.39..140471.53 rows=43119 width=69)
25	Hash Cond: (vessels_1.type = vesseltypes_1.code)
26	-> Hash Join (cost=18.00..140350.80 rows=43119 width=69)
27	Hash Cond: ((positions_1.vessel_id)::text = (vessels_1.id)::text)
28	-> Parallel Seq Scan on positions positions_1 (cost=0.00..140218.63 rows=43119 width=65)
29	Filter: (speed > '30'::numeric)
30	-> Hash (cost=11.89..11.89 rows=489 width=69)
31	-> Seq Scan on vessels vessels_1 (cost=0.00..11.89 rows=489 width=69)
32	-> Hash (cost=2.06..2.06 rows=106 width=4)
33	-> Seq Scan on vesseltypes vesseltypes_1 (cost=0.00..2.06 rows=106 width=4)
34	-> Sort (cost=5.63..5.89 rows=106 width=34)
35	Sort Key: vesseltypes.code
36	-> Seq Scan on vesseltypes (cost=0.00..2.06 rows=106 width=34)

Total rows: 36 of 36 | Query complete 00:00:00.065

iv.

```
12  SET max_parallel_workers_per_gather = 1024;
13  EXPLAIN
14  SELECT date(t), COUNT(lon) AS stigmas
15  FROM positions, vessels, vesseltypes
16  WHERE (date(t) BETWEEN '2019-08-14' and '2019-08-18')
17  AND (vessel_id = vessels.id AND type = code AND description LIKE 'Passenger%')
18  GROUP BY date(t)
19  ORDER BY date(t);
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	GroupAggregate (cost=151855.72..152315.15 rows=3319 width=12)
2	Group Key: (date(positions.t))
3	-> Gather Merge (cost=151855.72..152257.07 rows=3319 width=12)
4	Workers Planned: 5
5	-> Sort (cost=150855.64..150857.30 rows=664 width=12)
6	Sort Key: (date(positions.t))
7	-> Hash Join (cost=16.24..150824.52 rows=664 width=12)
8	Hash Cond: ((positions.vessel_id)::text = (vessels.id)::text)
9	-> Parallel Seq Scan on positions (cost=0.00..150773.60 rows=7037 width=81)
10	Filter: ((date(t) >= '2019-08-14'::date) AND (date(t) <= '2019-08-18'::date))
11	-> Hash (cost=15.67..15.67 rows=46 width=65)
12	-> Hash Join (cost=2.45..15.67 rows=46 width=65)
13	Hash Cond: (vessels.type = vesseltypes.code)
14	-> Seq Scan on vessels (cost=0.00..11.89 rows=489 width=69)
15	-> Hash (cost=2.33..2.33 rows=10 width=4)
16	-> Seq Scan on vesseltypes (cost=0.00..2.33 rows=10 width=4)
17	Filter: ((description)::text ~~ 'Passenger%'::text)

Total rows: 17 of 17 | Query complete 00:00:00.088

v.

```
12  SET max_parallel_workers_per_gather = 1024;
13  EXPLAIN
14  WITH question1(answer1) AS (
15      SELECT Distinct(vessel_id)
16      FROM positions, vessels, vesseltypes
17      WHERE (date(t) BETWEEN '2019-08-15' and '2019-08-18')
18      AND (vessel_id = vessels.id AND type = code AND description LIKE 'Cargo%')
19      ),
20  totalspeeds(ships, distance) AS (
21      SELECT vessel_id, SUM(speed)
22      FROM positions
23      WHERE (date(t) BETWEEN '2019-08-12' AND '2019-08-19')
24      GROUP BY vessel_id
25      ),
26  question2(answer2) AS (
27      SELECT DISTINCT(vessel_id)
28      FROM positions, vessels, vesseltypes, totalspeeds
29      WHERE vessel_id = ships AND distance = 0
30      AND (vessel_id = vessels.id AND type = code AND description LIKE 'Cargo%')
31      )
32  SELECT answer1 as stopped_once, answer2 as remained_stoped
33  FROM question1, question2;
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Nested Loop (cost=516801.85..519392.84 rows=180625 width=130)
2	-> Unique (cost=152240.31..152548.30 rows=425 width=65)
3	-> Gather Merge (cost=152240.31..152542.99 rows=2125 width=65)
4	Workers Planned: 5
5	-> Unique (cost=151240.23..151287.01 rows=425 width=65)
6	-> Merge Join (cost=151240.23..151285.35 rows=664 width=65)
7	Merge Cond: ((positions.vessel_id)::text = (vessels.id)::text)
8	-> Sort (cost=151223.29..151240.89 rows=7037 width=65)
9	Sort Key: positions.vessel_id
10	-> Parallel Seq Scan on positions (cost=0.00..150773.60 rows=7037 width=65)
11	Filter: ((date(t) >= '2019-08-15'::date) AND (date(t) <= '2019-08-18'::date))
12	-> Sort (cost=16.94..17.05 rows=46 width=65)
13	Sort Key: vessels.id
14	-> Hash Join (cost=2.45..15.67 rows=46 width=65)
15	Hash Cond: (vessels.type = vesseltypes.code)
16	-> Seq Scan on vessels (cost=0.00..11.89 rows=489 width=69)
17	-> Hash (cost=2.33..2.33 rows=10 width=4)
18	-> Seq Scan on vesseltypes (cost=0.00..2.33 rows=10 width=4)
19	Filter: ((description)::text ~~ 'Cargo%'::text)
20	-> Materialize (cost=364561.54..364583.54 rows=425 width=65)
21	-> Unique (cost=364561.54..364577.16 rows=425 width=65)
22	-> Sort (cost=364561.54..364569.35 rows=3124 width=65)
23	Sort Key: positions_1.vessel_id

24	-> Hash Join (cost=152575.97..364380.21 rows=3124 width=65)
25	Hash Cond: ((positions_1.vessel_id)::text = (vessels_1.id)::text)
26	-> Hash Join (cost=152559.72..364208.64 rows=33114 width=130)
27	Hash Cond: ((positions_1.vessel_id)::text = (totalspeeds.ships)::text)
28	-> Seq Scan on positions positions_1 (cost=0.00..192993.51 rows=7036651 width=65)
29	-> Hash (cost=152559.70..152559.70 rows=2 width=65)
30	-> Subquery Scan on totalspeeds (cost=152223.37..152559.70 rows=2 width=65)
31	-> Finalize GroupAggregate (cost=152223.37..152559.68 rows=2 width=97)
32	Group Key: positions_2.vessel_id
33	Filter: (sum(positions_2.speed) = '0'::numeric)
34	-> Gather Merge (cost=152223.37..152537.36 rows=2125 width=97)
35	Workers Planned: 5
36	-> Partial GroupAggregate (cost=151223.29..151281.38 rows=425 width=97)
37	Group Key: positions_2.vessel_id
38	-> Sort (cost=151223.29..151240.89 rows=7037 width=69)
39	Sort Key: positions_2.vessel_id
40	-> Parallel Seq Scan on positions positions_2 (cost=0.00..150773.60 rows=7037 width=69)
41	Filter: ((date(t) >= '2019-08-12'::date) AND (date(t) <= '2019-08-19'::date))
42	-> Hash (cost=15.67..15.67 rows=46 width=65)
43	-> Hash Join (cost=2.45..15.67 rows=46 width=65)
44	Hash Cond: (vessels_1.type = vesseltypes_1.code)
45	-> Seq Scan on vessels vessels_1 (cost=0.00..11.89 rows=489 width=69)
46	-> Hash (cost=2.33..2.33 rows=10 width=4)
47	-> Seq Scan on vesseltypes vesseltypes_1 (cost=0.00..2.33 rows=10 width=4)
48	Filter: ((description)::text ~~ 'Cargo%'::text)
Total rows: 48 of 48	Query complete 00:00:00.104

Παρατηρώ ότι έχουν αυξηθεί τα ‘workers’ που χρησιμοποιούνται για κάθε ερώτημα. Workers, είναι τα εργαλεία του προγράμματος για να την επεξεργασία δεδομένων και εκτέλεση διαδικασιών. Δίνοντας στο πρόγραμμα άδεια να χρησιμοποιήσει όσα workers όσα μπορεί να υποστηρίξει ο υπολογιστής, επιτρέπει στο πρόγραμμα να εκτελεί τις δουλειές που μπορεί να προκύψουν από ένα πρόβλημα ταυτόχρονα (αντί να πρέπει να ολοκληρώσει μία πριν αρχίσει την άλλη), βελτιώνοντας την απόδοση του προγράμματος και μειώνοντας το ‘Execution Time’.

Ερώτημα 4

i. Επειδή αυτό το επερώτημα δεν χρησιμοποιεί 'WHERE', δεν μπορεί να βοηθηθεί από ευρετήρια.

ii. Χρησιμοποιώ HASH Index καθώς είναι το πιο αποτελεσματικό για 'equality operations'. Παρόλα αυτά, το πρόγραμμα δεν το χρησιμοποίησε καθώς η συνθήκη του 'WHERE' ήταν αρκετά απλή και δεν θα βοηθιότανε από κάποιο Index.

```
5  CREATE INDEX index1 ON vessels USING HASH(flag);
6
7  explain analyze
8  SELECT type, COUNT(id) AS greek_ships
9  FROM vessels
10 where flag = 'Greece'
11 GROUP BY type
12 ORDER BY type;
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Sort (cost=15.49..15.58 rows=33 width=12) (actual time=0.178..0.179 rows=29 loops=1)
2	Sort Key: type
3	Sort Method: quicksort Memory: 26kB
4	-> HashAggregate (cost=14.33..14.66 rows=33 width=12) (actual time=0.159..0.164 rows=29 loops=1)
5	Group Key: type
6	Batches: 1 Memory Usage: 24kB
7	-> Seq Scan on vessels (cost=0.00..13.11 rows=244 width=69) (actual time=0.015..0.080 rows=244 loops=1)
8	Filter: ((flag)::text = 'Greece'::text)
9	Rows Removed by Filter: 245
10	Planning Time: 0.342 ms
11	Execution Time: 0.206 ms

Total rows: 11 of 11 | Query complete 00:00:00.044 | Rows selected: 1

iii. Χρησιμοποιώ B-TREE Index καθώς είναι το πιο αποτελεσματικό για συγκρίσεις. Βάζω μόνο το positions.speed στο ευρετήριο καθώς χρησιμοποιείται δύο φορές και όλες οι άλλες στήλες συγκρίνονται με στήλες από διαφορετικούς πίνακες, άρα δεν μπορεί να βοηθήσει αποτελεσματικά ένα ευρετήριο. Επίσης, έχοντας μόνο μία στήλη στο ευρετήριο δίνει καλύτερη απόδοση από ένα ευρετήριο με πολλές στήλες. Το ευρετήριο χρησιμοποιείται στην αναζήτηση 'speed > 30'. Επειδή το ευρετήριο είναι ταξινομημένο σε binary tree, οι αναζητήσεις σε αυτό γίνονται πιο γρήγορα. Οπότε, το πρόγραμμα αναζητάει στο ευρετήριο αντί για τον πίνακα 'positions' και μαζεύει τους δείκτες των σειρών που ψάχνει.

```
3  CREATE INDEX index1 ON positions using btree(speed ASC);
4
5  explain analyze
6  with shipcount(ship_type, total) as (
7      select type, count(distinct(vessel_id))
8      from positions, vessels, vesseltypes
9      where speed > 30 and vessel_id = vessels.id and type = code
10     group by type
11     )
12 select distinct(vessel_id), type, description, total as total_of_ships_by_type
13 from positions, vessels, vesseltypes, shipcount
14 where speed > 30 and vessel_id = vessels.id and type = code and code = ship_type;
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Unique (cost=404017.51..406491.98 rows=197958 width=107) (actual time=772.371..918.815 rows=16 loops=1)
2	-> Sort (cost=404017.51..404512.40 rows=197958 width=107) (actual time=772.369..881.379 rows=209734 loops=1)
3	Sort Key: positions.vessel_id, vessels.type, vesseltypes.description, (count(DISTINCT positions_1.vessel_id))
4	Sort Method: external merge Disk: 28360kB
5	-> Hash Join (cost=198252.06..375097.82 rows=197958 width=107) (actual time=455.898..586.131 rows=209734 loops=1)
6	Hash Cond: ((positions.vessel_id)::text = (vessels.id)::text)
7	-> Gather (cost=5043.29..179100.98 rows=215594 width=65) (actual time=29.273..102.576 rows=209741 loops=1)
8	Workers Planned: 2
9	Workers Launched: 2
10	-> Parallel Bitmap Heap Scan on positions (cost=4043.29..156541.58 rows=89831 width=65) (actual time=9.625..129.952 rows=69914 loops=3)
11	Recheck Cond: (speed > '30'::numeric)
12	Heap Blocks: exact=5827
13	-> Bitmap Index Scan on index1 (cost=0.00..3989.39 rows=215594 width=0) (actual time=21.961..21.961 rows=209741 loops=1)
14	Index Cond: (speed > '30'::numeric)
15	-> Hash (cost=193203.16..193203.16 rows=449 width=107) (actual time=426.604..427.420 rows=276 loops=1)
16	Buckets: 1024 Batches: 1 Memory Usage: 46kB
17	-> Merge Join (cost=169160.98..193203.16 rows=449 width=107) (actual time=226.304..427.338 rows=276 loops=1)
18	Merge Cond: (vessels.type = vesseltypes.code)
19	-> Merge Join (cost=169155.35..195350.48 rows=449 width=81) (actual time=226.249..427.209 rows=276 loops=1)
20	Merge Cond: (vessels.type = vessels_1.type)

```
21                    ->  Sort  (cost=33.73..34.96 rows=489 width=69) (actual time=0.296..0.334 rows=405 loops=1)
22                        Sort Key: vessels.type
23                        Sort Method: quicksort  Memory: 78kB
24                        ->  Seq Scan on vessels  (cost=0.00..11.89 rows=489 width=69) (actual time=0.022..0.102 rows=489 loops=1)
25                    ->  GroupAggregate  (cost=169121.62..195309.40 rows=33 width=12) (actual time=225.945..426.791 rows=9 loops=1)
26                        Group Key: vessels_1.type
27                        ->  Gather Merge  (cost=169121.62..194231.10 rows=215594 width=69) (actual time=225.904..285.749 rows=209734 loops=1)
28                            Workers Planned: 2
29                            Workers Launched: 2
30                            ->  Sort  (cost=168121.60..168346.18 rows=89831 width=69) (actual time=159.246..170.796 rows=69911 loops=3)
31                                Sort Key: vessels_1.type
32                                Sort Method: external merge  Disk: 8160kB
33                                Worker 0:  Sort Method: external merge  Disk: 2488kB
34                                Worker 1:  Sort Method: external merge  Disk: 5792kB
35                                ->  Hash Join  (cost=4064.67..157045.29 rows=89831 width=69) (actual time=12.228..131.885 rows=69911 loops=3)
36                                    Hash Cond: (vessels_1.type = vesseltypes_1.code)
37                                    ->  Hash Join  (cost=4061.29..156797.44 rows=89831 width=69) (actual time=12.014..114.963 rows=69914 loops=3)
38                                        Hash Cond: ((positions_1.vessel_id)::text = (vessels_1.id)::text)
39                                        ->  Parallel Bitmap Heap Scan on positions positions_1  (cost=4043.29..156541.58 rows=89831 width=65) (actual time=11.758..97.048 rows=69914
40                                            Recheck Cond: (speed > '30'::numeric)
41                                            Heap Blocks: exact=17564
42                                            ->  Bitmap Index Scan on index1  (cost=0.00..3989.39 rows=215594 width=0) (actual time=28.300..28.300 rows=209741 loops=1)
43                                                Index Cond: (speed > '30'::numeric)
44                                        ->  Hash  (cost=11.89..11.89 rows=489 width=69) (actual time=0.188..0.189 rows=489 loops=3)
45                                            Buckets: 1024  Batches: 1  Memory Usage: 57kB
46                                            ->  Seq Scan on vessels vessels_1  (cost=0.00..11.89 rows=489 width=69) (actual time=0.027..0.092 rows=489 loops=3)
47                                    ->  Hash  (cost=2.06..2.06 rows=106 width=4) (actual time=0.064..0.064 rows=106 loops=3)
48                                        Buckets: 1024  Batches: 1  Memory Usage: 12kB
49                                        ->  Seq Scan on vesseltypes vesseltypes_1  (cost=0.00..2.06 rows=106 width=4) (actual time=0.037..0.043 rows=106 loops=3)
50                    ->  Sort  (cost=5.63..5.89 rows=106 width=34) (actual time=0.048..0.070 rows=81 loops=1)
51                        Sort Key: vesseltypes.code
52                        Sort Method: quicksort  Memory: 33kB
53                        ->  Seq Scan on vesseltypes  (cost=0.00..2.06 rows=106 width=34) (actual time=0.020..0.028 rows=106 loops=1)
54    Planning Time: 0.888 ms
55    Execution Time: 922.620 ms
```

Total rows: 55 of 55 | Query complete 00:00:00.976

iv. Χρησιμοποιώ B-TREE Index καθώς είναι το πιο αποτελεσματικό για συγκρίσεις (BETWEEN). Βάζω το date(positions.t) στο ευρετήριο καθώς τα στοιχεία της στήλη 't' καλούνται μόνο στην μορφή date(t). Επειδή το ευρετήριο είναι σε μορφή binary tree, οι αναζητήσεις σε αυτό γίνονται πιο γρήγορα. Οπότε, το πρόγραμμα αναζητάει στο ευρετήριο αντί για τον πίνακα 'positions' και μαζεύει τους δείκτες των σειρών που ψάχνει. Όλες οι άλλες στήλες συγκρίνονται με στήλες από διαφορετικούς πίνακες, άρα δεν μπορεί να βοηθήσει αποτελεσματικά ένα ευρετήριο. Η στήλη 'vesseltypes.description' εξαιρείται, καθώς θα μπορούσα να φτιάξω δεύτερο ευρετήριο για αυτήν, το πρόγραμμα δεν θα το χρησιμοποιούσε για τους ίδιους λόγους με το δεύτερο υποερώτημα.

```
3  CREATE INDEX index1 ON positions using btree(date(t));
4
5  explain analyze
6  SELECT date(t), COUNT(lon) AS stigmas
7  FROM positions, vessels, vesseltypes
8  WHERE (date(t) BETWEEN '2019-08-14' and '2019-08-18')
9  AND (vessel_id = vessels.id AND type = code AND description LIKE 'Passenger%')
10 GROUP BY date(t)
11 ORDER BY date(t);
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	GroupAggregate (cost=78408.23..78474.61 rows=3319 width=12) (actual time=333.701..377.063 rows=5 loops=1)
2	Group Key: (date(positions.t))
3	-> Sort (cost=78408.23..78416.52 rows=3319 width=12) (actual time=330.877..353.484 rows=361493 loops=1)
4	Sort Key: (date(positions.t))
5	Sort Method: external merge Disk: 8064kB
6	-> Hash Join (cost=497.30..78214.12 rows=3319 width=12) (actual time=34.623..280.253 rows=361493 loops=1)
7	Hash Cond: ((positions.vessel_id)::text = (vessels.id)::text)
8	-> Bitmap Heap Scan on positions (cost=481.06..78024.55 rows=35183 width=81) (actual time=34.500..117.135 rows=1177212 loops=1)
9	Recheck Cond: ((date(t) >= '2019-08-14'::date) AND (date(t) <= '2019-08-18'::date))
10	Heap Blocks: exact=20719
11	-> Bitmap Index Scan on index1 (cost=0.00..472.26 rows=35183 width=0) (actual time=30.902..30.903 rows=1177212 loops=1)
12	Index Cond: ((date(t) >= '2019-08-14'::date) AND (date(t) <= '2019-08-18'::date))
13	-> Hash (cost=15.67..15.67 rows=46 width=65) (actual time=0.111..0.114 rows=64 loops=1)
14	Buckets: 1024 Batches: 1 Memory Usage: 15kB
15	-> Hash Join (cost=2.45..15.67 rows=46 width=65) (actual time=0.028..0.103 rows=64 loops=1)
16	Hash Cond: (vessels.type = vesseltypes.code)
17	-> Seq Scan on vessels (cost=0.00..11.89 rows=489 width=69) (actual time=0.006..0.033 rows=489 loops=1)
18	-> Hash (cost=2.33..2.33 rows=10 width=4) (actual time=0.016..0.017 rows=10 loops=1)
19	Buckets: 1024 Batches: 1 Memory Usage: 9kB
20	-> Seq Scan on vesseltypes (cost=0.00..2.33 rows=10 width=4) (actual time=0.010..0.013 rows=10 loops=1)
21	Filter: ((description)::text ~~ 'Passenger%'::text)
22	Rows Removed by Filter: 96
23	Planning Time: 0.745 ms
24	Execution Time: 378.415 ms

Total rows: 24 of 24 | Query complete 00:00:00.414

v. Χρησιμοποιώ B-TREE Index καθώς είναι το πιο αποτελεσματικό για συγκρίσεις (BETWEEN). Βάζω το date(positions.t) στο ευρετήριο καθώς τα στοιχεία της στήλη ‘t’ καλούνται μόνο στην μορφή date(t). Επειδή το ευρετήριο είναι σε μορφή binary tree, οι αναζητήσεις σε αυτό γίνονται πιο γρήγορα. Οπότε, το πρόγραμμα αναζητάει στο ευρετήριο αντί για τον πίνακα ‘positions’ και μαζεύει τους δείκτες των σειρών που ψάχνει. Όλες οι άλλες στήλες συγκρίνονται με στήλες από διαφορετικούς πίνακες, άρα δεν μπορεί να βοηθήσει αποτελεσματικά ένα ευρετήριο. Η στήλη ‘vesseltypes.description’ εξαιρείται, καθώς θα μπορούσα να φτιάξω δεύτερο ευρετήριο για αυτήν, το πρόγραμμα δεν θα το χρησιμοποιούσε για τους ίδιους λόγους με το δεύτερο και τέταρτο υποερώτημα.

```sql
CREATE INDEX index1 ON positions using btree(date(t));

explain analyze
WITH question1(answer1) AS (
    SELECT Distinct(vessel_id)
    FROM positions, vessels, vesseltypes
    WHERE (date(t) BETWEEN '2019-08-15' and '2019-08-18')
    AND (vessel_id = vessels.id AND type = code AND description LIKE 'Cargo%')
    ),
totalspeeds(ships, distance) AS (
    SELECT vessel_id, SUM(speed)
    FROM positions
    WHERE (date(t) BETWEEN '2019-08-12' AND '2019-08-19')
    GROUP BY vessel_id
    ),
question2(answer2) AS (
    SELECT DISTINCT(vessel_id)
    FROM positions, vessels, vesseltypes, totalspeeds
    WHERE vessel_id = ships AND distance = 0
    AND (vessel_id = vessels.id AND type = code AND description LIKE 'Cargo%')
    )
SELECT answer1 as stopped_once, answer2 as remained_stoped
FROM question1, question2;
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Nested Loop (cost=318292.00..320767.92 rows=180625 width=130) (actual time=1818.718..1867.697 rows=39 loops=1)
2	-> Unique (cost=78399.93..78416.52 rows=425 width=65) (actual time=305.456..335.261 rows=39 loops=1)
3	-> Sort (cost=78399.93..78408.23 rows=3319 width=65) (actual time=305.455..329.942 rows=68540 loops=1)
4	Sort Key: positions.vessel_id
5	Sort Method: external merge Disk: 5032kB
6	-> Hash Join (cost=497.30..78205.82 rows=3319 width=65) (actual time=32.158..243.530 rows=68540 loops=1)
7	Hash Cond: ((positions.vessel_id)::text = (vessels.id)::text)
8	-> Bitmap Heap Scan on positions (cost=481.06..78024.55 rows=35183 width=65) (actual time=31.786..120.007 rows=1104218 loops=1)
9	Recheck Cond: ((date(t) >= '2019-08-15'::date) AND (date(t) <= '2019-08-18'::date))
10	Heap Blocks: exact=19444
11	-> Bitmap Index Scan on index1 (cost=0.00..472.26 rows=35183 width=0) (actual time=28.815..28.815 rows=1104218 loops=1)
12	Index Cond: ((date(t) >= '2019-08-15'::date) AND (date(t) <= '2019-08-18'::date))
13	-> Hash (cost=15.67..15.67 rows=46 width=65) (actual time=0.205..0.210 rows=104 loops=1)
14	Buckets: 1024 Batches: 1 Memory Usage: 18kB
15	-> Hash Join (cost=2.45..15.67 rows=46 width=65) (actual time=0.041..0.183 rows=104 loops=1)
16	Hash Cond: (vessels.type = vesseltypes.code)
17	-> Seq Scan on vessels (cost=0.00..11.89 rows=489 width=69) (actual time=0.009..0.049 rows=489 loops=1)
18	-> Hash (cost=2.33..2.33 rows=10 width=4) (actual time=0.025..0.028 rows=10 loops=1)
19	Buckets: 1024 Batches: 1 Memory Usage: 9kB
20	-> Seq Scan on vesseltypes (cost=0.00..2.33 rows=10 width=4) (actual time=0.014..0.019 rows=10 loops=1)
21	Filter: ((description)::text ~~ 'Cargo%'::text)
22	Rows Removed by Filter: 96
23	-> Materialize (cost=239892.07..240090.40 rows=425 width=65) (actual time=38.802..39.292 rows=1 loops=39)

```
24  -> Unique  (cost=239892.07..240084.02 rows=425 width=65) (actual time=1513.234..1532.372 rows=1 loops=1)
25    -> Gather Merge  (cost=239892.07..240081.90 rows=850 width=65) (actual time=1513.233..1532.368 rows=3 loops=1)
26          Workers Planned: 2
27          Workers Launched: 2
28          -> Unique  (cost=238892.05..238983.76 rows=425 width=65) (actual time=1470.415..1485.312 rows=1 loops=3)
29                -> Merge Join  (cost=238892.05..238980.51 rows=1302 width=65) (actual time=1470.414..1483.751 rows=21117 loops=3)
30                      Merge Cond: ((positions_1.vessel_id)::text = (vessels_1.id)::text)
31                      -> Sort  (cost=238875.11..238909.61 rows=13798 width=130) (actual time=1469.919..1471.930 rows=21181 loops=3)
32                            Sort Key: positions_1.vessel_id
33                            Sort Method: external merge  Disk: 3384kB
34                            Worker 0:  Sort Method: quicksort  Memory: 3646kB
35                            Worker 1:  Sort Method: quicksort  Memory: 4021kB
36                            -> Hash Join  (cost=78206.88..237926.35 rows=13798 width=130) (actual time=833.410..1461.791 rows=21181 loops=3)
37                                  Hash Cond: ((positions_1.vessel_id)::text = (totalspeeds.ships)::text)
38                                  -> Parallel Seq Scan on positions positions_1  (cost=0.00..151946.38 rows=2931938 width=65) (actual time=0.033..349.338 rows=2345550 loops=3)
39                                  -> Hash  (cost=78206.86..78206.86 rows=2 width=65) (actual time=833.357..833.360 rows=3 loops=3)
40                                        Buckets: 1024  Batches: 1  Memory Usage: 9kB
41                                        -> Subquery Scan on totalspeeds  (cost=78200.46..78206.86 rows=2 width=65) (actual time=833.281..833.346 rows=3 loops=3)
42                                              -> HashAggregate  (cost=78200.46..78206.84 rows=2 width=97) (actual time=833.280..833.344 rows=3 loops=3)
43                                                    Group Key: positions_2.vessel_id
44                                                    Filter: (sum(positions_2.speed) = '0'::numeric)
45                                                    Batches: 1  Memory Usage: 285kB
46                                                    Rows Removed by Filter: 339
47                                                    Worker 0:  Batches: 1  Memory Usage: 285kB
48                                                    Worker 1:  Batches: 1  Memory Usage: 285kB
49                                                    -> Bitmap Heap Scan on positions positions_2  (cost=481.06..78024.55 rows=35183 width=69) (actual time=51.405..253.036 rows=1592303 lo...
50                                                          Recheck Cond: ((date(t) >= '2019-08-12'::date) AND (date(t) <= '2019-08-19'::date))
51                                                          Heap Blocks: exact=28061
52                                                          -> Bitmap Index Scan on index1  (cost=0.00..472.26 rows=35183 width=0) (actual time=46.431..46.431 rows=1592303 loops=3)
53                                                                Index Cond: ((date(t) >= '2019-08-12'::date) AND (date(t) <= '2019-08-19'::date))
54                      -> Sort  (cost=16.94..17.05 rows=46 width=65) (actual time=0.469..1.235 rows=21211 loops=3)
55                            Sort Key: vessels_1.id
56                            Sort Method: quicksort  Memory: 36kB
57                            Worker 0:  Sort Method: quicksort  Memory: 36kB
58                            Worker 1:  Sort Method: quicksort  Memory: 36kB
59                            -> Hash Join  (cost=2.45..15.67 rows=46 width=65) (actual time=0.175..0.280 rows=104 loops=3)
60                                  Hash Cond: (vessels_1.type = vesseltypes_1.code)
61                                  -> Seq Scan on vessels vessels_1  (cost=0.00..11.89 rows=489 width=69) (actual time=0.077..0.115 rows=489 loops=3)
62                                  -> Hash  (cost=2.33..2.33 rows=10 width=4) (actual time=0.052..0.053 rows=10 loops=3)
63                                        Buckets: 1024  Batches: 1  Memory Usage: 9kB
64                                        -> Seq Scan on vesseltypes vesseltypes_1  (cost=0.00..2.33 rows=10 width=4) (actual time=0.043..0.046 rows=10 loops=3)
65                                              Filter: ((description)::text ~~ 'Cargo%'::text)
66                                              Rows Removed by Filter: 96
67  Planning Time: 1.767 ms
68  Execution Time: 1869.319 ms
```

Total rows: 68 of 68 | Query complete 00:00:01.953 | Rows selected: 2