Overview of all tables:

All the data is auto-generated, so some names/types/containers/... do not make too much sense.

1.Nation: this table includes information about the countries mentioned in this database. n\_nationkey is a unique key to every country. n\_regionkey can be 0, 1, 2, 3, 4. ⇒ corresponds to r\_regionkey. n\_comment is the comment about that country.

schema:

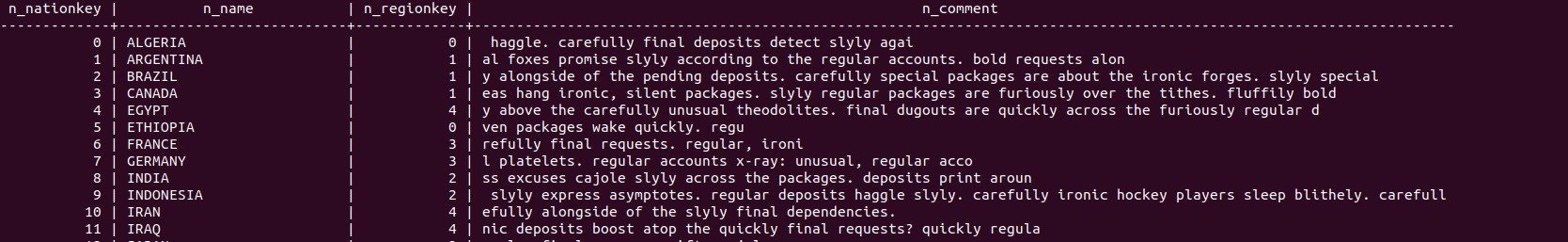
CREATE TABLE NATION ( N\_NATIONKEY INTEGER NOT NULL,

N\_NAME CHAR(25) NOT NULL,

N\_REGIONKEY INTEGER NOT NULL,

N\_COMMENT VARCHAR(152));

Example:



2. Region: this table includes information about the regions mentioned in this database. There are only five of them.

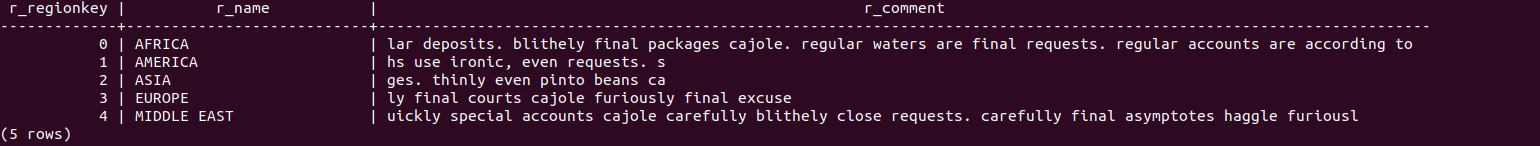
schema:

CREATE TABLE REGION ( R\_REGIONKEY INTEGER NOT NULL,

R\_NAME CHAR(25) NOT NULL,

R\_COMMENT VARCHAR(152));

Example:



3. Part: this table includes information about items, or parts of a product.

schema:

CREATE TABLE PART ( P\_PARTKEY INTEGER NOT NULL,

P\_NAME VARCHAR(55) NOT NULL,

P\_MFGR CHAR(25) NOT NULL,

P\_BRAND CHAR(10) NOT NULL,

P\_TYPE VARCHAR(25) NOT NULL,

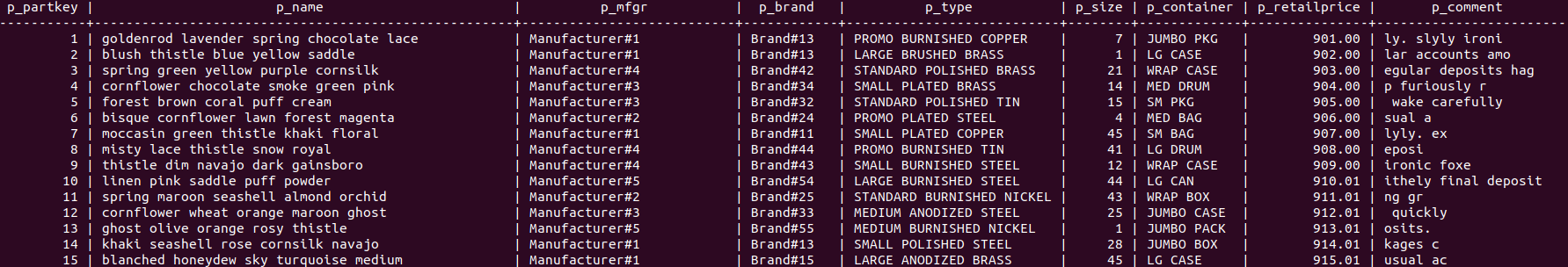
P\_SIZE INTEGER NOT NULL,

P\_CONTAINER CHAR(10) NOT NULL,

P\_RETAILPRICE DECIMAL(15,2) NOT NULL,

P\_COMMENT VARCHAR(23) NOT NULL );

Example:



4. Supplier: this table includes information about suppliers around the world.

s\_accbal: supplier account balance.

schema:

CREATE TABLE SUPPLIER ( S\_SUPPKEY INTEGER NOT NULL,

S\_NAME CHAR(25) NOT NULL,

S\_ADDRESS VARCHAR(40) NOT NULL,

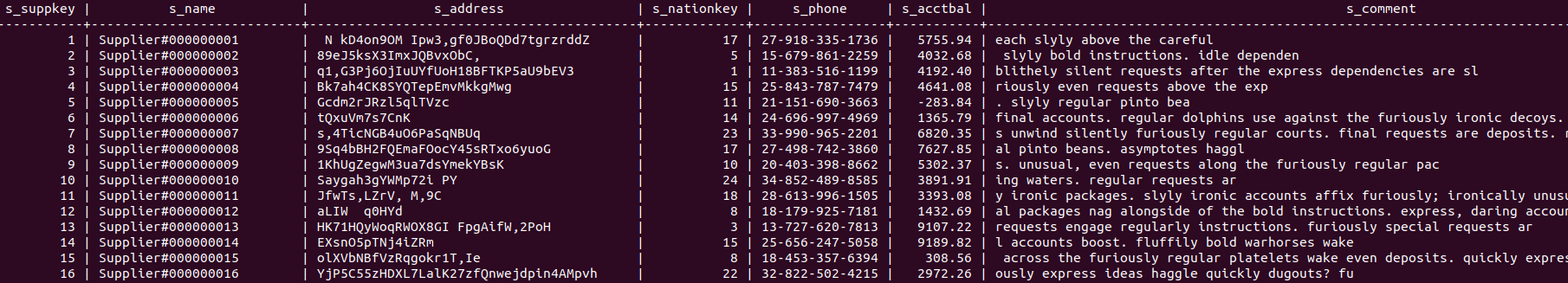
S\_NATIONKEY INTEGER NOT NULL,

S\_PHONE CHAR(15) NOT NULL,

S\_ACCTBAL DECIMAL(15,2) NOT NULL,

S\_COMMENT VARCHAR(101) NOT NULL);

Example:



5. Partsupp: this table includes information about parts, their suppliers, available quantity, supply cost, and comments on that part supply.

schema:

CREATE TABLE PARTSUPP ( PS\_PARTKEY INTEGER NOT NULL,

PS\_SUPPKEY INTEGER NOT NULL,

PS\_AVAILQTY INTEGER NOT NULL,

PS\_SUPPLYCOST DECIMAL(15,2) NOT NULL,

PS\_COMMENT VARCHAR(199) NOT NULL );

Example:

6. Customer: this table includes information about customers.

schema:

CREATE TABLE CUSTOMER ( C\_CUSTKEY INTEGER NOT NULL,

C\_NAME VARCHAR(25) NOT NULL,

C\_ADDRESS VARCHAR(40) NOT NULL,

C\_NATIONKEY INTEGER NOT NULL,

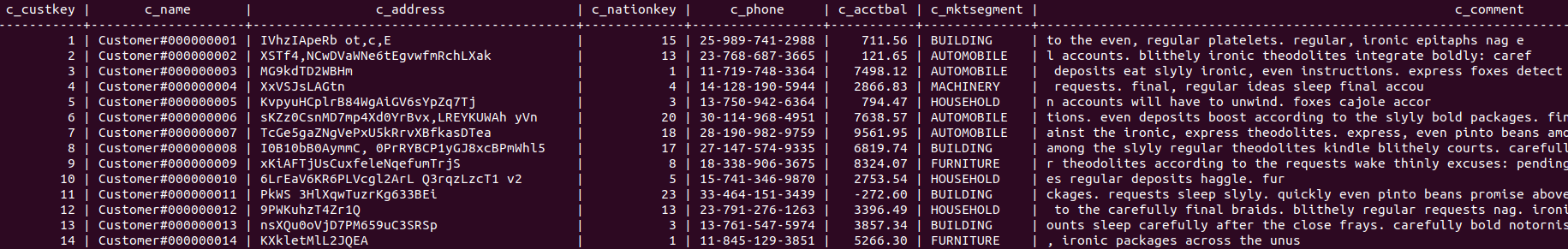
C\_PHONE CHAR(15) NOT NULL,

C\_ACCTBAL DECIMAL(15,2) NOT NULL,

C\_MKTSEGMENT CHAR(10) NOT NULL,

C\_COMMENT VARCHAR(117) NOT NULL);

Example:



7. Orders: contains information about all orders. O\_orderstatus: ‘F’ means failed to meet the committed delivery date.

schema:

CREATE TABLE ORDERS ( O\_ORDERKEY INTEGER NOT NULL,

O\_CUSTKEY INTEGER NOT NULL,

O\_ORDERSTATUS CHAR(1) NOT NULL,

O\_TOTALPRICE DECIMAL(15,2) NOT NULL,

O\_ORDERDATE DATE NOT NULL,

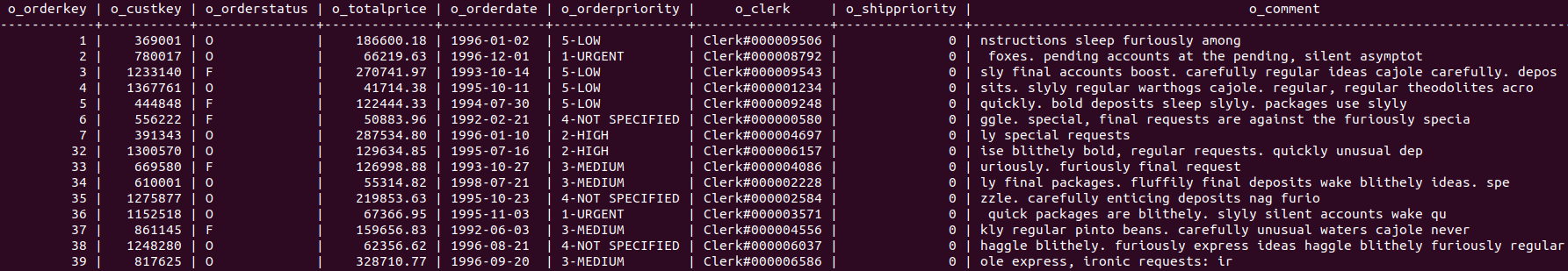
O\_ORDERPRIORITY CHAR(15) NOT NULL,

O\_CLERK CHAR(15) NOT NULL,

O\_SHIPPRIORITY INTEGER NOT NULL,

O\_COMMENT VARCHAR(79) NOT NULL);

Example:



8. Lineitem: contains information about all line items(in transportation). L\_ORDERKEY: unique key of an order.

schema:

CREATE TABLE LINEITEM ( L\_ORDERKEY INTEGER NOT NULL,

L\_PARTKEY INTEGER NOT NULL,

L\_SUPPKEY INTEGER NOT NULL,

L\_LINENUMBER INTEGER NOT NULL,

L\_QUANTITY DECIMAL(15,2) NOT NULL,

L\_EXTENDEDPRICE DECIMAL(15,2) NOT NULL,

L\_DISCOUNT DECIMAL(15,2) NOT NULL,

L\_TAX DECIMAL(15,2) NOT NULL,

L\_RETURNFLAG CHAR(1) NOT NULL,

L\_LINESTATUS CHAR(1) NOT NULL,

L\_SHIPDATE DATE NOT NULL,

L\_COMMITDATE DATE NOT NULL,

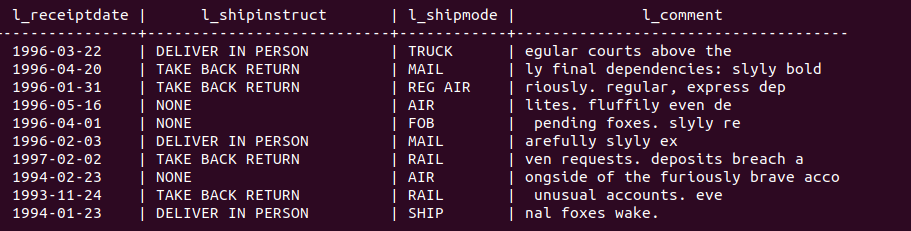
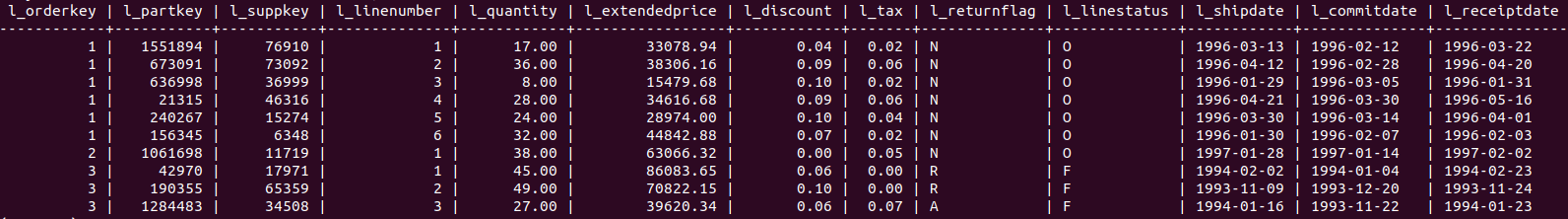
L\_RECEIPTDATE DATE NOT NULL,

L\_SHIPINSTRUCT CHAR(25) NOT NULL,

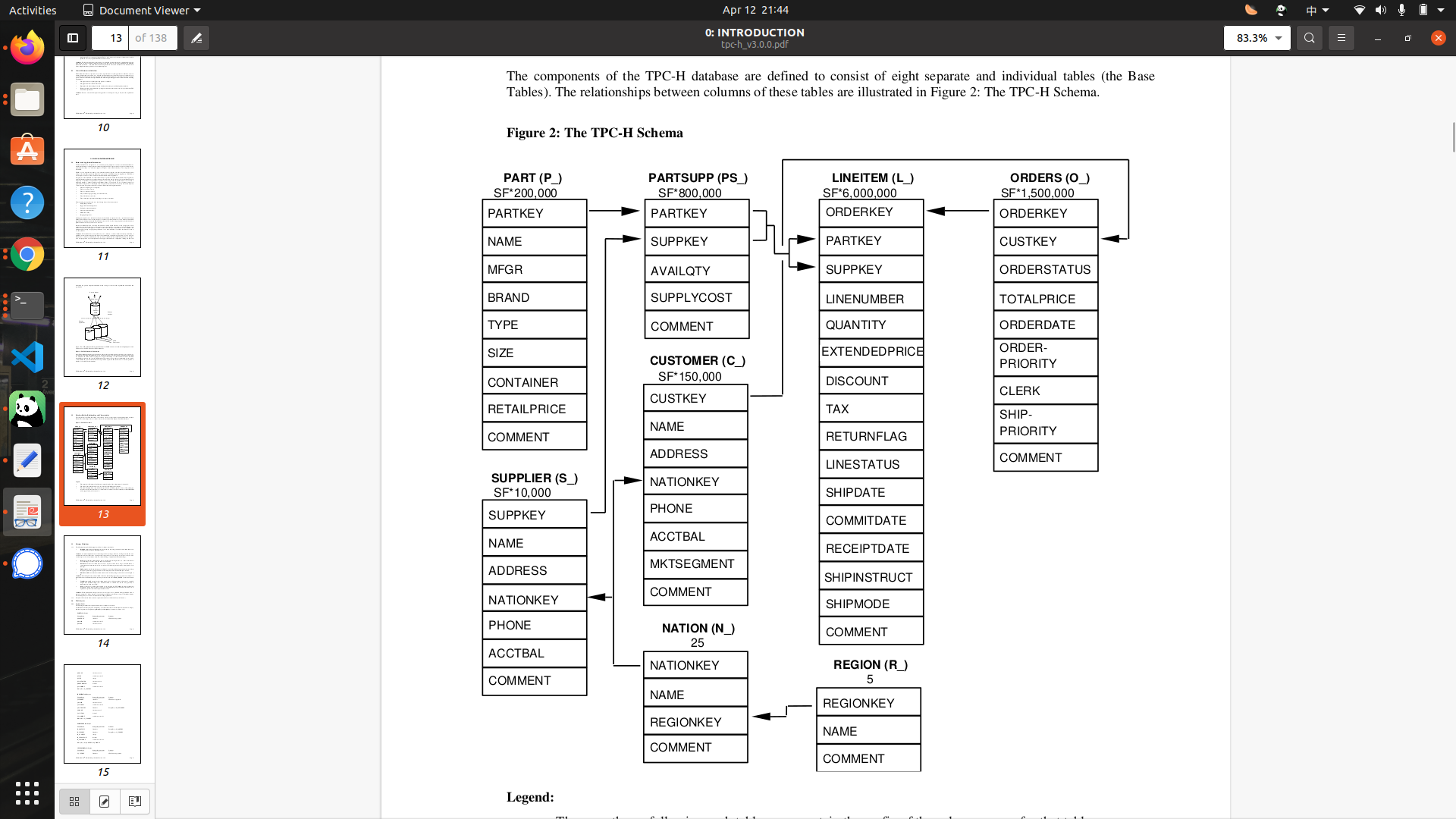
L\_SHIPMODE CHAR(10) NOT NULL,

L\_COMMENT VARCHAR(44) NOT NULL);

Example:



**Overall Schema**



**Problem 1.**

**Parts/Supplier Relationship Query (Example 16)**

This query finds out how many suppliers can supply parts with given attributes.

Table used: **partsupp, part**

Query broad description:

The Parts/Supplier Relationship Query **counts the number of (distinct) suppliers** who can supply parts that satisfy a particular customer's requirements. The customer is interested in parts of **eight different sizes** as long as **they are not of a given type[Brand#45]**, **not of a given brand[MEDIUM POLISHED]**, and **not from a supplier who has bad complaints** .

Results must be presented in

First, **descending count** (of the number of suppliers)

Second, **ascending brand**

Third, **ascending** **type**

Fourth, **ascending** **size**.

Specific Requirements:

1. BRAND is **not** Brand#45.

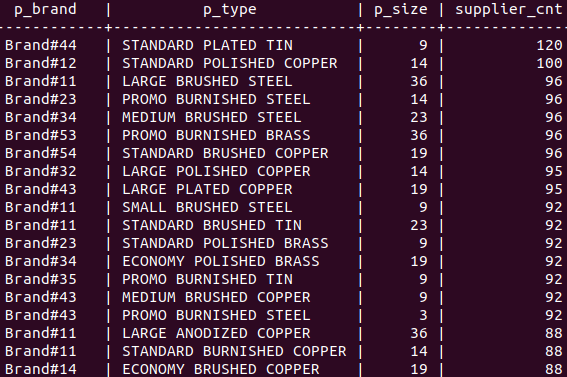
2. TYPE does **not** start with **MEDIUM POLISHED** .

3. Size must be one of {49, 14, 23, 45, 19, 3, 36, 9}

4. comment of supplier must not contain “Customer Complaints”

In other words, s\_comment should not match '%Customer%Complaints%'

Example of output table:



**Problem 2.**

**Forecasting Revenue Change Query (Example 6)**

This query quantifies the amount of revenue increase that would have resulted from eliminating certain company-wide discounts in a given percentage range in a given year. Asking this type of "what if" query can be used to look for ways to increase revenues.

Table used: **lineitem**

Query broad description:

The Forecasting Revenue Change Query considers all the lineitems shipped **in a given year “1994-01-01” - “1995-01-01”** with **discounts between 0.05 and 0.07**. The query lists the amount by which the total revenue would have increased if these discounts had been eliminated for lineitems **with l\_quantity less than 24**. Note that the potential revenue increase is equal to t**he sum of [l\_extendedprice \* l\_discount]** for all lineitems with discounts and quantities in the qualifying range.

Sample output:

revenue

-----------------

1230113636.0101

3677125983.5401

**Problem 3.**

**Order Priority Checking Query (Example 4)**

This query determines how well the order priority system is working and gives an assessment of customer satisfaction.

Table used: orders, lineitem

Query broad description:

The Order Priority Checking Query counts the number of orders ordered in a given quarter of a given year (“1993-07-01” to “1993-10-01”) in which **at least one lineitem was received by the customer later than its committed date**. The query lists the count of such orders for each order priority sorted in ascending priority order.

Sample output:

o\_orderpriority | order\_count

-----------------------------+-------------

1-URGENT | 105214

2-HIGH | 104821

3-MEDIUM | 105227

4-NOT SPECIFIED | 105422

5-LOW | 105356

For benchmark 16(first 1)

**Understanding 1:**

Groundtruth is:

select

p\_brand,

p\_type,

p\_size,

count(distinct ps\_suppkey) as supplier\_cnt

from

partsupp,

part

where

p\_partkey = ps\_partkey

and p\_brand <> 'Brand#45'

and p\_type not like 'MEDIUM POLISHED%'

and p\_size in (49, 14, 23, 45, 19, 3, 36, 9)

and ps\_suppkey not in (

select

s\_suppkey

from

supplier

where

s\_comment like '%Customer%Complaints%'

)

group by

p\_brand,

p\_type,

p\_size

order by

supplier\_cnt desc,

p\_brand,

p\_type,

p\_size;

Running time & Query plan for ground truth is:

QUERY PLAN

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Sort (cost=379739.94..380123.78 rows=153537 width=44) (actual time=2239.933..2266.070 rows=27840 loops=1)

Sort Key: (count(DISTINCT partsupp.ps\_suppkey)) DESC, part.p\_brand, part.p\_type, part.p\_size

Sort Method: quicksort Memory: 2968kB

-> GroupAggregate (cost=285383.54..361789.08 rows=153537 width=44) (actual time=1692.061..2219.225 rows=27840 loops=1)

Group Key: part.p\_brand, part.p\_type, part.p\_size

-> Gather Merge (cost=285383.54..354333.55 rows=592016 width=40) (actual time=1692.018..2029.149 rows=1186602 loops=1)

Workers Planned: 2

Workers Launched: 2

-> Sort (cost=284383.51..285000.19 rows=246673 width=40) (actual time=1671.124..1791.961 rows=395534 loops=3)

Sort Key: part.p\_brand, part.p\_type, part.p\_size

Sort Method: external merge Disk: 20144kB

Worker 0: Sort Method: external merge Disk: 20256kB

Worker 1: Sort Method: external merge Disk: 20200kB

-> Parallel Hash Join (cost=67767.50..255543.18 rows=246673 width=40) (actual time=824.326..1034.508 rows=395534 loops=3)

Hash Cond: (partsupp.ps\_partkey = part.p\_partkey)

-> Parallel Index Only Scan using partsupp\_pkey on partsupp (cost=3466.46..172881.12 rows=1666667 width=8) (actual time=23.633..439.421 rows=2665173 loops=3)

Filter: (NOT (hashed SubPlan 1))

Rows Removed by Filter: 1493

Heap Fetches: 0

SubPlan 1

-> Seq Scan on supplier (cost=0.00..3466.00 rows=10 width=4) (actual time=2.586..23.174 rows=56 loops=3)

Filter: ((s\_comment)::text ~~ '%Customer%Complaints%'::text)

Rows Removed by Filter: 99944

-> Parallel Hash (cost=61795.33..61795.33 rows=123337 width=40) (actual time=128.485..128.486 rows=98941 loops=3)

Buckets: 65536 Batches: 8 Memory Usage: 3328kB

-> Parallel Seq Scan on part (cost=0.00..61795.33 rows=123337 width=40) (actual time=8.477..111.524 rows=98941 loops=3)

Filter: ((p\_brand <> 'Brand#45'::bpchar) AND ((p\_type)::text !~~ 'MEDIUM POLISHED%'::text) AND (p\_size = ANY ('{49,14,23,45,19,3,36,9}'::integer[])))

Rows Removed by Filter: 567725

Planning Time: 1.075 ms

JIT:

Functions: 84

Options: Inlining false, Optimization false, Expressions true, Deforming true

Timing: Generation 8.631 ms, Inlining 0.000 ms, Optimization 1.614 ms, Emission 30.807 ms, Total 41.052 ms

Execution Time: 2273.876 ms

(34 rows)

Handwritten solution is:

SELECT

p\_brand,

p\_type,

p\_size,

count(distinct ps\_suppkey) AS SUPPLIERCNT

FROM

(SELECT p\_brand, p\_type, p\_size, p\_partkey, ps\_comment, ps\_suppkey

FROM PART RIGHT OUTER JOIN PARTSUPP

ON part.p\_partkey = partsupp.ps\_partkey) as a

WHERE

p\_brand <> 'Brand#45'

AND p\_type NOT LIKE 'MEDIUM POLISHED%'

AND (p\_size = 49 OR p\_size = 14 OR p\_size = 23 OR p\_size = 45

OR p\_size = 19 OR p\_size = 3 OR p\_size = 36 OR p\_size = 9)

AND ps\_suppkey not in (

select

s\_suppkey

from

supplier

where

s\_comment like '%Customer%Complaints%'

)

GROUP BY p\_brand, p\_type, p\_size

ORDER BY SUPPLIERCNT DESC, p\_brand, p\_type, p\_size;

Running time & Query plan for Handwritten solution is:

QUERY PLAN

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Sort (cost=380210.97..380583.71 rows=149095 width=44) (actual time=2303.700..2329.026 rows=27840 loops=1)

Sort Key: (count(DISTINCT partsupp.ps\_suppkey)) DESC, part.p\_brand, part.p\_type, part.p\_size

Sort Method: quicksort Memory: 2968kB

-> GroupAggregate (cost=291445.64..362810.83 rows=149095 width=44) (actual time=1728.399..2278.697 rows=27840 loops=1)

Group Key: part.p\_brand, part.p\_type, part.p\_size

-> Gather Merge (cost=291445.64..355794.76 rows=552512 width=40) (actual time=1728.361..2081.984 rows=1186602 loops=1)

Workers Planned: 2

Workers Launched: 2

-> Sort (cost=290445.61..291021.14 rows=230213 width=40) (actual time=1714.437..1843.689 rows=395534 loops=3)

Sort Key: part.p\_brand, part.p\_type, part.p\_size

Sort Method: external merge Disk: 20424kB

Worker 0: Sort Method: external merge Disk: 20080kB

Worker 1: Sort Method: external merge Disk: 20096kB

-> Parallel Hash Join (cost=75933.96..263645.64 rows=230213 width=40) (actual time=844.255..1058.498 rows=395534 loops=3)

Hash Cond: (partsupp.ps\_partkey = part.p\_partkey)

-> Parallel Index Only Scan using partsupp\_pkey on partsupp (cost=3466.46..172881.12 rows=1666667 width=8) (actual time=17.824..449.094 rows=2665173 loops=3)

Filter: (NOT (hashed SubPlan 1))

Rows Removed by Filter: 1493

Heap Fetches: 0

SubPlan 1

-> Seq Scan on supplier (cost=0.00..3466.00 rows=10 width=4) (actual time=2.522..17.396 rows=56 loops=3)

Filter: ((s\_comment)::text ~~ '%Customer%Complaints%'::text)

Rows Removed by Filter: 99944

-> Parallel Hash (cost=70128.67..70128.67 rows=115107 width=40) (actual time=128.577..128.577 rows=98941 loops=3)

Buckets: 65536 Batches: 8 Memory Usage: 3360kB

-> Parallel Seq Scan on part (cost=0.00..70128.67 rows=115107 width=40) (actual time=9.390..111.653 rows=98941 loops=3)

Filter: ((p\_brand <> 'Brand#45'::bpchar) AND ((p\_type)::text !~~ 'MEDIUM POLISHED%'::text) AND ((p\_size = 49) OR (p\_size = 14) OR (p\_size = 23) OR (p\_size = 45) OR (p\_size = 19) OR (p\_size = 3) OR (p\_size = 36) OR (p\_size = 9)))

Rows Removed by Filter: 567725

Planning Time: 1.123 ms

JIT:

Functions: 84

Options: Inlining false, Optimization false, Expressions true, Deforming true

Timing: Generation 8.629 ms, Inlining 0.000 ms, Optimization 1.810 ms, Emission 33.220 ms, Total 43.659 ms

Execution Time: 2336.719 ms

(34 rows)

**Understanding 2:**

Groundtruth is:

**select**

**p\_brand,**

**p\_type,**

**p\_size,**

**count(distinct ps\_suppkey) as supplier\_cnt**

**from**

**partsupp,**

**part**

**where**

**p\_partkey = ps\_partkey**

**and p\_brand <> 'Brand#45'**

**and p\_type not like 'MEDIUM POLISHED%'**

**and p\_size in (49, 14, 23, 45, 19, 3, 36, 9)**

**and ps\_comment not like '%Customer%Complaints%'**

**group by**

**p\_brand,**

**p\_type,**

**p\_size**

**order by**

**supplier\_cnt desc,**

**p\_brand,**

**p\_type,**

**p\_size;**

Running time & Query plan for Groundtruth is:

QUERY PLAN

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Sort (cost=546454.05..546837.89 rows=153537 width=44) (actual time=2640.381..2667.860 rows=27840 loops=1)

Sort Key: (count(DISTINCT partsupp.ps\_suppkey)) DESC, part.p\_brand, part.p\_type, part.p\_size

Sort Method: quicksort Memory: 2968kB

-> GroupAggregate (cost=377242.39..528503.20 rows=153537 width=44) (actual time=2084.004..2617.965 rows=27840 loops=1)

Group Key: part.p\_brand, part.p\_type, part.p\_size

-> Gather Merge (cost=377242.39..515128.69 rows=1183914 width=40) (actual time=2083.953..2436.535 rows=1187296 loops=1)

Workers Planned: 2

Workers Launched: 2

-> Sort (cost=376242.37..377475.61 rows=493298 width=40) (actual time=2062.641..2189.700 rows=395765 loops=3)

Sort Key: part.p\_brand, part.p\_type, part.p\_size

Sort Method: external merge Disk: 20264kB

Worker 0: Sort Method: external merge Disk: 20104kB

Worker 1: Sort Method: external merge Disk: 20272kB

-> Parallel Hash Join (cost=64301.05..316106.86 rows=493298 width=40) (actual time=1147.348..1364.936 rows=395765 loops=3)

Hash Cond: (partsupp.ps\_partkey = part.p\_partkey)

-> Parallel Seq Scan on partsupp (cost=0.00..216052.67 rows=3333000 width=8) (actual time=0.018..670.743 rows=2666667 loops=3)

Filter: ((ps\_comment)::text !~~ '%Customer%Complaints%'::text)

-> Parallel Hash (cost=61795.33..61795.33 rows=123337 width=40) (actual time=205.911..205.911 rows=98941 loops=3)

Buckets: 65536 Batches: 8 Memory Usage: 3328kB

-> Parallel Seq Scan on part (cost=0.00..61795.33 rows=123337 width=40) (actual time=92.040..188.547 rows=98941 loops=3)

Filter: ((p\_brand <> 'Brand#45'::bpchar) AND ((p\_type)::text !~~ 'MEDIUM POLISHED%'::text) AND (p\_size = ANY ('{49,14,23,45,19,3,36,9}'::integer[])))

Rows Removed by Filter: 567725

Planning Time: 0.911 ms

JIT:

Functions: 45

Options: Inlining true, Optimization true, Expressions true, Deforming true

Timing: Generation 6.459 ms, Inlining 61.440 ms, Optimization 130.582 ms, Emission 83.621 ms, Total 282.102 ms

Execution Time: 2674.829 ms

(28 rows)

Handwritten solution is:

**SELECT**

**p\_brand,**

**p\_type,**

**p\_size,**

**count(distinct ps\_suppkey) AS SUPPLIERCNT**

**FROM**

**(SELECT p\_brand, p\_type, p\_size, p\_partkey, ps\_comment, ps\_suppkey**

**FROM PART RIGHT OUTER JOIN PARTSUPP**

**ON part.p\_partkey = partsupp.ps\_partkey) as a**

**WHERE**

**p\_brand <> 'Brand#45'**

**AND p\_type NOT LIKE 'MEDIUM POLISHED%'**

**AND (p\_size = 49 OR p\_size = 14 OR p\_size = 23 OR p\_size = 45**

**OR p\_size = 19 OR p\_size = 3 OR p\_size = 36 OR p\_size = 9)**

**AND ps\_comment NOT LIKE '%Customer%Complaints%'**

**GROUP BY p\_brand, p\_type, p\_size**

**ORDER BY SUPPLIERCNT DESC, p\_brand , p\_type , p\_size ;**

Running time & Query plan for Handwritten solution is:

QUERY PLAN

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Sort (cost=539728.80..540101.54 rows=149095 width=44) (actual time=2612.695..2640.125 rows=27840 loops=1)

Sort Key: (count(DISTINCT partsupp.ps\_suppkey)) DESC, part.p\_brand, part.p\_type, part.p\_size

Sort Method: quicksort Memory: 2968kB

-> GroupAggregate (cost=381103.26..522328.66 rows=149095 width=44) (actual time=2052.271..2589.690 rows=27840 loops=1)

Group Key: part.p\_brand, part.p\_type, part.p\_size

-> Gather Merge (cost=381103.26..509788.58 rows=1104913 width=40) (actual time=2052.237..2408.973 rows=1187296 loops=1)

Workers Planned: 2

Workers Launched: 2

-> Sort (cost=380103.23..381254.18 rows=460380 width=40) (actual time=2027.657..2154.565 rows=395765 loops=3)

Sort Key: part.p\_brand, part.p\_type, part.p\_size

Sort Method: external merge Disk: 20376kB

Worker 0: Sort Method: external merge Disk: 20272kB

Worker 1: Sort Method: external merge Disk: 19984kB

-> Parallel Hash Join (cost=72467.50..324209.32 rows=460380 width=40) (actual time=1130.760..1342.282 rows=395765 loops=3)

Hash Cond: (partsupp.ps\_partkey = part.p\_partkey)

-> Parallel Seq Scan on partsupp (cost=0.00..216052.67 rows=3333000 width=8) (actual time=0.020..663.662 rows=2666667 loops=3)

Filter: ((ps\_comment)::text !~~ '%Customer%Complaints%'::text)

-> Parallel Hash (cost=70128.67..70128.67 rows=115107 width=40) (actual time=197.905..197.906 rows=98941 loops=3)

Buckets: 65536 Batches: 8 Memory Usage: 3296kB

-> Parallel Seq Scan on part (cost=0.00..70128.67 rows=115107 width=40) (actual time=101.863..180.627 rows=98941 loops=3)

Filter: ((p\_brand <> 'Brand#45'::bpchar) AND ((p\_type)::text !~~ 'MEDIUM POLISHED%'::text) AND ((p\_size = 49) OR (p\_size = 14) OR (p\_size = 23) OR (p\_size = 45) OR (p\_size = 19) OR (p\_size = 3) OR (p\_size = 36) OR (p\_size = 9)))

Rows Removed by Filter: 567725

Planning Time: 0.219 ms

JIT:

Functions: 45

Options: Inlining true, Optimization true, Expressions true, Deforming true

Timing: Generation 3.095 ms, Inlining 61.253 ms, Optimization 144.282 ms, Emission 99.417 ms, Total 308.046 ms

Execution Time: 2644.084 ms