ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΜΑΤΙΚΗΣ

Ομάδα 17

Δαδιώτης Δημήτριος - 216108

Ιωάννης Σκουλούδης - 2021091

Περιεχόμενα:

Ερώτημα 1

1) Το μέγεθος του αρχείου αν είναι αρχείο σωρού.

Μέγεθος εγγραφής: 200 byte

Μέγεθος block: 1024 byte

Bfr = (B/R) = 1024 / 200 = 5 εγγραφές ανά Block

b=ceiling(r/bfr)=ceiling(15.000.000 / 5) = **3.000.000 blocks** ~ 3.072 GB

2) Το μέγεθος του αρχείου αν είναι αρχείο κατακερματισμού.

Υποθέτουμε πως το αρχείο κατακερματισμού είναι 75% γεμάτο

b=ceiling(r/bfr)*125/100= ceiling(15.000.000 / 5)*125/100 = **3.750.000 blocks** ~3.84 GB

3) Επίπεδα B* δένδρου.

Εφόσον πρόκειται για αταξινόμητο αρχείο και χτίζουμε το ευρετήριο σε πεδίο με διπλότυπα , θα χτίσουμε πυκνό ευρετήριο και θα έχουμε μια εγγραφή ευρετηρίου για κάθε εγγραφή του αρχείου.

V=20 bytes (Μήκος πεδίου κλειδιού αναζήτησης)

Pr=12 bytes (Μέγεθος δείκτη εγγραφής)

P=16 bytes (Μέγεθος δείκτη block)

B=1024 bytes (Μέγεθος block)

$(p * P) + ((p - 1) * V) \leq B \Rightarrow (p * 16) + (p - 1) * 20 \leq B \Rightarrow 16p + 20p - 20 \leq 1024 \Rightarrow$

$36 * p \leq 1044 \Rightarrow p \leq 29$

Άρα p = 29

$P_{leaf} * (P_r + V) + P \leq B \Rightarrow P_{leaf} *(12 + 20) + 16 \leq 1024 \Rightarrow 32 * P_{leaf} + 16 \leq 1024 \Rightarrow$

$32 * P_{leaf} \leq 1008 \Rightarrow P_{leaf} \leq 31.5$

Παίρνουμε το floor οπότε $P_{leaf} = 31$

Άρα σε ένα block χωράνε 31 ζευγάρια κλειδιού 20 bytes και Pointer P_r των 16 bytes

Έτσι βρίσκω πόσα block έχω στο τελευταίο επίπεδο των φύλλων 15.000.000 / 31 = 483.871

Τα B* δέντρα πρέπει να είναι τουλάχιστον 66% γεμάτα. Υποθέτουμε πως το δικό μας είναι 66% γεμάτο οπότε κατά μέσο όρο κάθε εσωτερικός κόμβος θα έχει 29 * 0.66 = 19 δείκτες οπότε 18 κλειδιά

Κάθε κόμβος φύλλο θα έχει 31 * 100% = 31 δείκτες εγγραφών δεδομένων.

Έχουμε το τελευταίο επίπεδο(φύλλα) 483.871 κόμβους

Επόμενο επίπεδο (5) 483.871 / 19 = 25.467 κόμβους

Επόμενο επίπεδο (4) 25.467 / 19 = 1.341 κόμβους

Επόμενο επίπεδο (3) 1.341 / 19 = 71 κόμβους

Επόμενο επίπεδο (2) 70 / 19 = 4 κόμβους

Επόμενο επίπεδο (1) 4 / 9= 1 κόμβος για την ρίζα

Άρα έχουμε **h = 6**

4) Κόμβοι ανά επίπεδο και μέγεθος ευρετηρίου .

Σύμφωνα με το 3 βλέπουμε τους κόμβους ανα επίπεδο και συνολικά το μέγεθος του ευρετηρίου θα είναι:

Συνολικά έχουμε 483.871+25.467+1.341+71+4+1 = **510.755 blocks**

5) B+ δέντρο κόμβοι ανά επίπεδο και μέγεθος ευρετηρίου.

Τα B+ δέντρα πρέπει να είναι τουλάχιστον 50% πλήρης, Υποθέτουμε πως το B+ δέντρο μας είναι 50% πλήρης. Οπότε έχουμε 29 * 0.5 = 14 δείκτες οπότε 13 κλειδιά και κάθε κόμβος φύλλο θα έχει 31 * 100% = 31 δείκτες εγγραφών δεδομένων.

Εχουμε το τελευταίο επίπεδο (φύλλα) 483.871 κόμβους

Επόμενο επίπεδο (5) 483.871 / 14 = 34.563 κόμβους

Επόμενο επίπεδο (4) 34.563 / 14 = 2.469 κόμβους

Επόμενο επίπεδο (3) 2.469 / 14 = 177 κόμβους

Επόμενο επίπεδο (2) 177 / 14 = 13 κόμβους

Επόμενο επίπεδο (1) 13 / 14 = 1 κόμβος όπου είναι η ρίζα

Άρα το μέγεθος του ευρετηρίου θα είναι:

Συνολικά έχουμε 483.871+ 34.563 + 2.469 + 177 + 13 + 1 = **521.094 blocks**

6) Ευρεση κόστους αναζήτησης ισότητας.

Το κόστος αναζήτησης ισότητας θα είναι: h + 20 + 1 = 6 + 20 + 1 = **27**

Ερώτημα 2

α) Πεδίο age_group.

```
create or replace function get_age_group(birth_date date) return VARCHAR is
age number(3);
age_group varchar(10);

BEGIN

  age := FLOOR(MONTHS_BETWEEN(SYSDATE, birth_date) / 12);


  IF age < 40 THEN
    age_group := 'Under 40';
  ELSIF age BETWEEN 40 AND 50 THEN
    age_group := '40-50';
  ELSIF age BETWEEN 50 AND 60 THEN
    age_group := '50-60';
  ELSIF age BETWEEN 60 AND 70 THEN
    age_group := '60-70';
  ELSE
    age_group := 'Above 70';
  END IF;

  RETURN age_group;
END get_age_group;
```

β) Πεδίο income_level.

```
CREATE OR REPLACE FUNCTION get_income_level(income_level VARCHAR2) RETURN VARCHAR2 IS
  first_number NUMBER;
  second_number NUMBER;
  max_income NUMBER;

BEGIN

  first_number:=TO_NUMBER(REGEXP_REPLACE(REGEXP_SUBSTR(income_level, '[0-9,]+', 1, 1), ',', ''));
  second_number:=TO_NUMBER(REGEXP_REPLACE(REGEXP_SUBSTR(income_level, '[0-9,]+', 1, 2), ',', ''));
  max_income:=GREATEST(first_number,COALESCE(second_number, 0));

  IF max_income <= 129999 THEN
    RETURN 'low';
  ELSIF max_income >129999 and max_income<= 249999 THEN
    RETURN 'medium';
  ELSIF max_income >= 250000 THEN
    RETURN 'high';
  ELSE
    RETURN 'other';
  END IF;
END get_income_level;
```

γ) Πεδίο marital_status.

```
CREATE OR REPLACE FUNCTION fix_status(marital_status VARCHAR2) RETURN VARCHAR2 IS
  fixed_status VARCHAR2(20);
  v_marital_status VARCHAR2(20);
BEGIN

  v_marital_status := UPPER(marital_status);

  IF v_marital_status IN ('WIDOWED', 'SEPAR.', 'DIVORCED', 'NEVERM', 'SINGLE', 'DIVORC.') THEN
    fixed_status := 'Single';
  ELSIF v_marital_status = 'MARRIED' THEN
    fixed_status := 'Married';
  ELSE
    fixed_status := 'Unknown';
  END IF;

  RETURN fixed_status;
END fix_status;
```

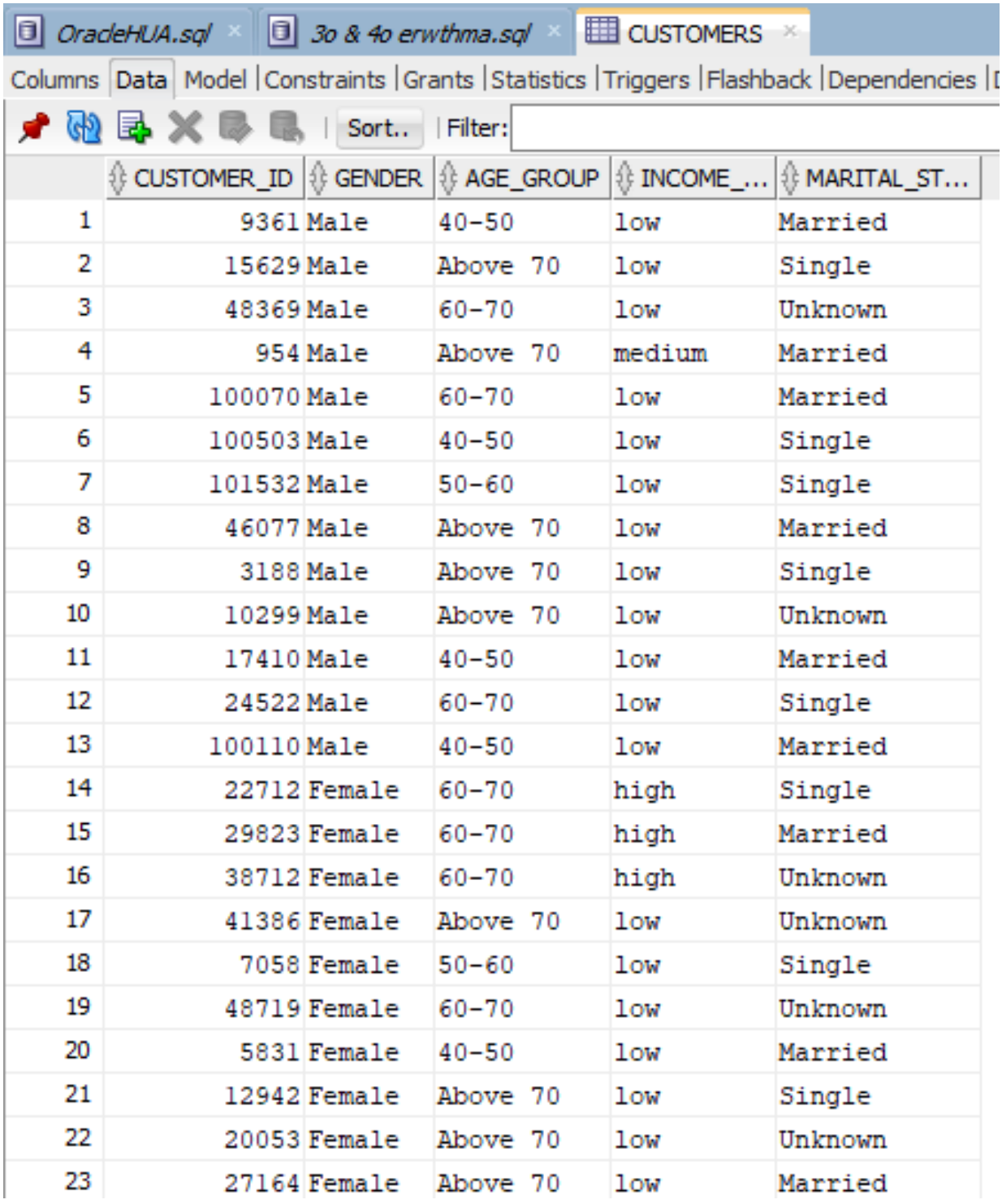
Παρακάτω φαίνεται η δημιουργία καθώς και το περιεχόμενο των tables CUSTOMERS, PRODUCTS και ORDERS

```
create table customers as
select id as customer_id ,
gender ,
get_age_group(birth_date) as age_group ,
fix_status(marital_status) as marital_status,
get_income_level(income_level) as income_level
from xsales.customers;


create table products as
select p.identifier as product_id , p.name as productname , c.name as categoryname ,p.list_price as list_price
from xsales.products p join xsales.categories c on  p.subcategory_reference=c.id;

alter session set NLS_DATE_FORMAT='DD/MM/YY ';

create table orders as
select o.id as order_id ,
oi.product_id as product_id ,
o.customer_id as customer_id ,
oi.amount as price ,
o.order_finished - oi.order_date as days_to_process,
oi.cost as cost,
o.channel as channel
from xsales.orders o join xsales.order_items oi on  o.id=oi.order_id;
```

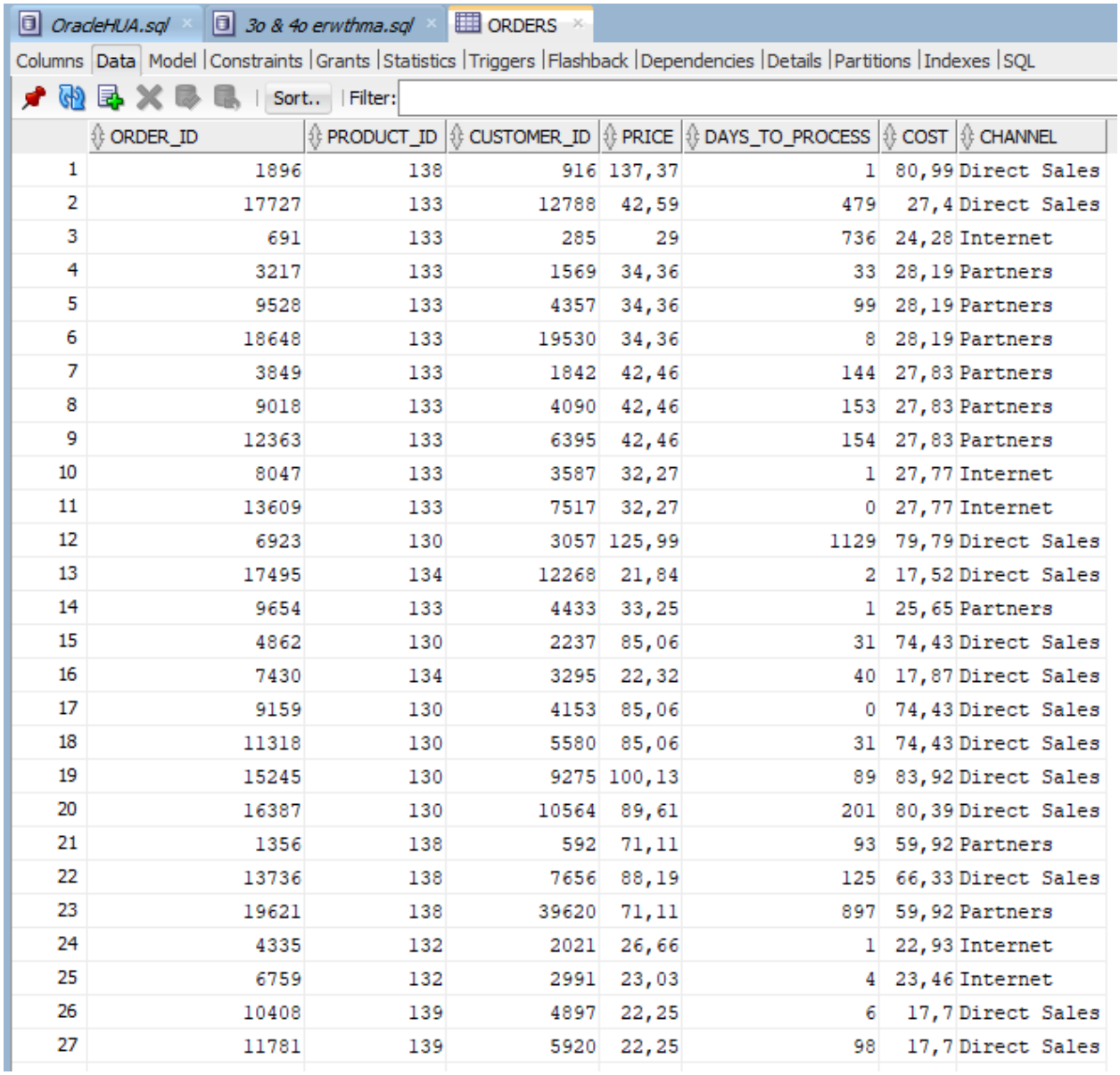
Script Output

Task completed in 5,581 seconds

```
Table CUSTOMERS created.


Table PRODUCTS created.


Session altered.


Table ORDERS created.
```



	CUSTOMER_ID	GENDER	AGE_GROUP	INCOME_...	MARITAL_ST...
1	9361	Male	40-50	low	Married
2	15629	Male	Above 70	low	Single
3	48369	Male	60-70	low	Unknown
4	954	Male	Above 70	medium	Married
5	100070	Male	60-70	low	Married
6	100503	Male	40-50	low	Single
7	101532	Male	50-60	low	Single
8	46077	Male	Above 70	low	Married
9	3188	Male	Above 70	low	Single
10	10299	Male	Above 70	low	Unknown
11	17410	Male	40-50	low	Married
12	24522	Male	60-70	low	Single
13	100110	Male	40-50	low	Married
14	22712	Female	60-70	high	Single
15	29823	Female	60-70	high	Married
16	38712	Female	60-70	high	Unknown
17	41386	Female	Above 70	low	Unknown
18	7058	Female	50-60	low	Single
19	48719	Female	60-70	low	Unknown
20	5831	Female	40-50	low	Married
21	12942	Female	Above 70	low	Single
22	20053	Female	Above 70	low	Unknown
23	27164	Female	Above 70	low	Married

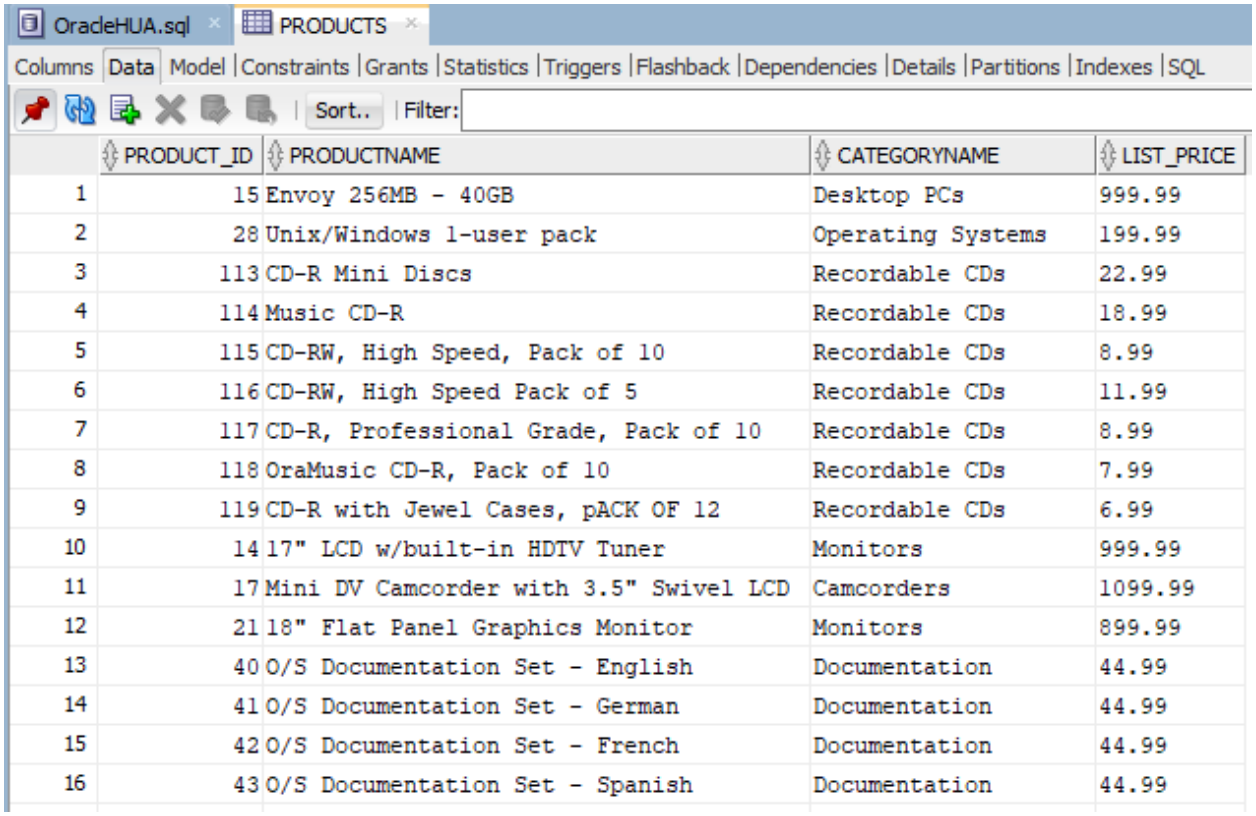


OracleHUA.sql | 3o & 4o erwthma.sql | ORDERS

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Sort.. | Filter:

	ORDER_ID	PRODUCT_ID	CUSTOMER_ID	PRICE	DAYS_TO_PROCESS	COST	CHANNEL
1	1896	138	916	137,37	1	80,99	Direct Sales
2	17727	133	12788	42,59	479	27,4	Direct Sales
3	691	133	285	29	736	24,28	Internet
4	3217	133	1569	34,36	33	28,19	Partners
5	9528	133	4357	34,36	99	28,19	Partners
6	18648	133	19530	34,36	8	28,19	Partners
7	3849	133	1842	42,46	144	27,83	Partners
8	9018	133	4090	42,46	153	27,83	Partners
9	12363	133	6395	42,46	154	27,83	Partners
10	8047	133	3587	32,27	1	27,77	Internet
11	13609	133	7517	32,27	0	27,77	Internet
12	6923	130	3057	125,99	1129	79,79	Direct Sales
13	17495	134	12268	21,84	2	17,52	Direct Sales
14	9654	133	4433	33,25	1	25,65	Partners
15	4862	130	2237	85,06	31	74,43	Direct Sales
16	7430	134	3295	22,32	40	17,87	Direct Sales
17	9159	130	4153	85,06	0	74,43	Direct Sales
18	11318	130	5580	85,06	31	74,43	Direct Sales
19	15245	130	9275	100,13	89	83,92	Direct Sales
20	16387	130	10564	89,61	201	80,39	Direct Sales
21	1356	138	592	71,11	93	59,92	Partners
22	13736	138	7656	88,19	125	66,33	Direct Sales
23	19621	138	39620	71,11	897	59,92	Partners
24	4335	132	2021	26,66	1	22,93	Internet
25	6759	132	2991	23,03	4	23,46	Internet
26	10408	139	4897	22,25	6	17,7	Direct Sales
27	11781	139	5920	22,25	98	17,7	Direct Sales



OracleHUA.sql | PRODUCTS

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Sort.. | Filter:

	PRODUCT_ID	PRODUCTNAME	CATEGORYNAME	LIST_PRICE
1	15	Envoy 256MB - 40GB	Desktop PCs	999.99
2	28	Unix/Windows 1-user pack	Operating Systems	199.99
3	113	CD-R Mini Discs	Recordable CDs	22.99
4	114	Music CD-R	Recordable CDs	18.99
5	115	CD-RW, High Speed, Pack of 10	Recordable CDs	8.99
6	116	CD-RW, High Speed Pack of 5	Recordable CDs	11.99
7	117	CD-R, Professional Grade, Pack of 10	Recordable CDs	8.99
8	118	OraMusic CD-R, Pack of 10	Recordable CDs	7.99
9	119	CD-R with Jewel Cases, pACK OF 12	Recordable CDs	6.99
10	14	17" LCD w/built-in HDTV Tuner	Monitors	999.99
11	17	Mini DV Camcorder with 3.5" Swivel LCD	Camcorders	1099.99
12	21	18" Flat Panel Graphics Monitor	Monitors	899.99
13	40	O/S Documentation Set - English	Documentation	44.99
14	41	O/S Documentation Set - German	Documentation	44.99
15	42	O/S Documentation Set - French	Documentation	44.99
16	43	O/S Documentation Set - Spanish	Documentation	44.99

i) Μέγιστη καθυστέρηση εκτέλεσης ανά παραγγελία.

```
create or replace function max_delay( id number) return number is

v_days_to_process number;

cursor c1 is
select max(days_to_process)
from orders
where order_id= id;
begin
    open c1;
    fetch c1 into v_days_to_process;
    close c1;

    if v_days_to_process>20 then
        v_days_to_process:=v_days_to_process-20;
    elsif v_days_to_process<=20 then
         v_days_to_process:=0;
    end if;

    return  v_days_to_process;
end;
```

Η παραπάνω συνάρτηση παίρνει ως όρισμα το order_id και επιστρέφει την μέγιστη καθυστέρηση της παραγγελίας.Για παράδειγμα για order_id 1,2,32,41,19 έχουμε:

```
BEGIN
    DBMS_OUTPUT.PUT_LINE(max_delay(1));
    DBMS_OUTPUT.PUT_LINE(max_delay(2));
    DBMS_OUTPUT.PUT_LINE(max_delay(32));
    DBMS_OUTPUT.PUT_LINE(max_delay(41));
    DBMS_OUTPUT.PUT_LINE(max_delay(19));
END;
/
```

Script Output

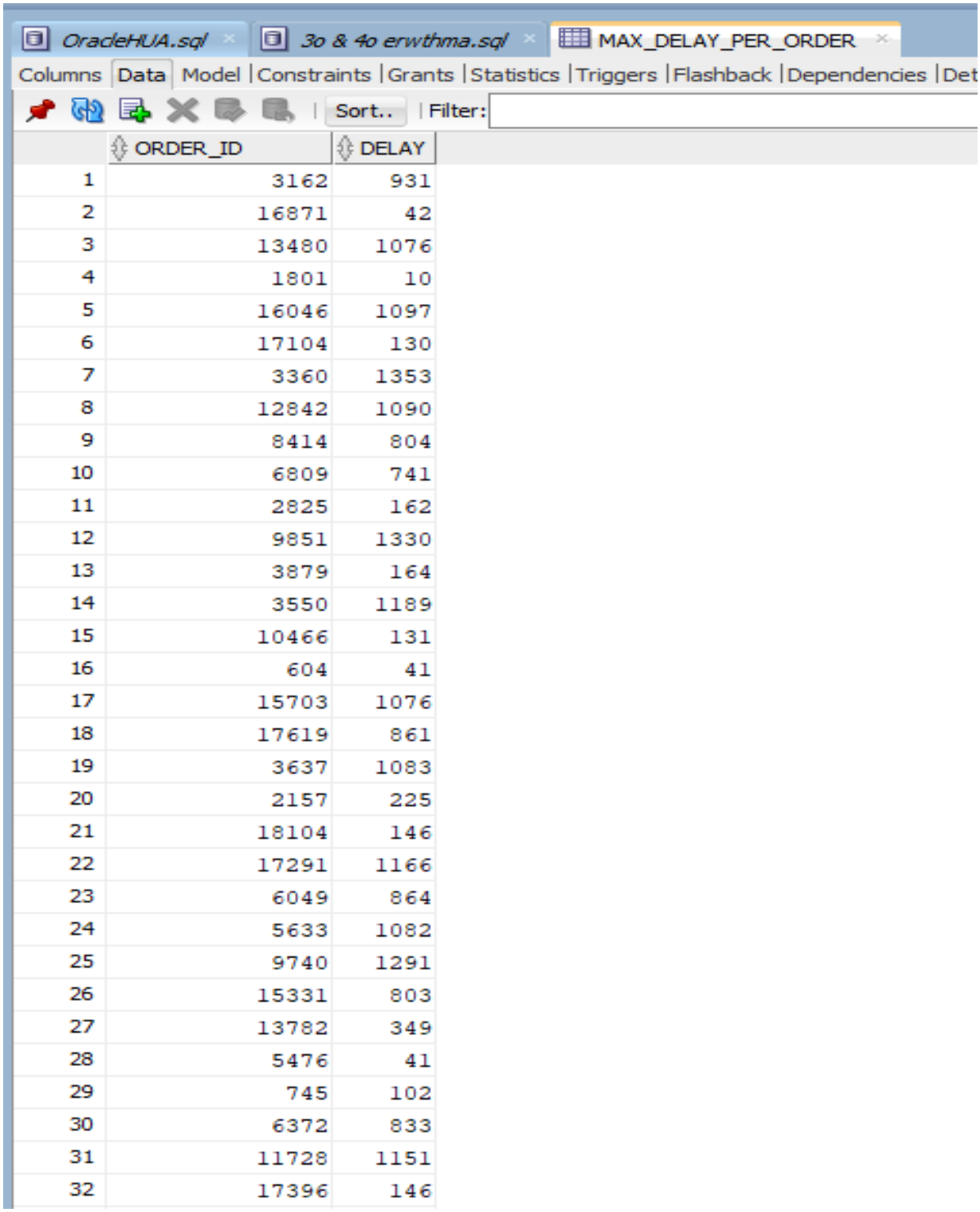
Task completed in 1,485 seconds

```
0
0
747
1107
355


PL/SQL procedure successfully completed.
```

Παρακάτω έχουμε την υλοποίηση μιας διεργασίας find_max_delay_per_order(δύο διαφορετικές υλοποιήσεις μια που χρησιμοποιεί την συνάρτηση και μια χωρίς) η οποία γεμίζει τον πίνακα max_delay_per_order με την καθυστέρηση κάθε παραγγελίας

```
create table max_delay_per_order(
order_id number,
delay number
);
```

```sql
create or replace procedure find_max_delay_per_order as

begin

    delete from max_delay_per_order;

    insert into max_delay_per_order
    select  distinct order_id, max_delay(order_id) as delay
    from orders;

end;

create or replace procedure find_max_delay_per_order as

begin

    delete from max_delay_per_order;

    insert into max_delay_per_order
    select order_id, max(days_to_process) - 20 as delay
    from orders
    where days_to_process > 20
    group by order_id

    union all

    select order_id, 0 as delay
    from orders
    group by order_id
    having max(days_to_process)<=20;

end;

execute find_max_delay_per_order;
```



	ORDER_ID	DELAY
1	3162	931
2	16871	42
3	13480	1076
4	1801	10
5	16046	1097
6	17104	130
7	3360	1353
8	12842	1090
9	8414	804
10	6809	741
11	2825	162
12	9851	1330
13	3879	164
14	3550	1189
15	10466	131
16	604	41
17	15703	1076
18	17619	861
19	3637	1083
20	2157	225
21	18104	146
22	17291	1166
23	6049	864
24	5633	1082
25	9740	1291
26	15331	803
27	13782	349
28	5476	41
29	745	102
30	6372	833
31	11728	1151
32	17396	146

ii) Τελικό κέρδος ανά παραγγελία.

```
create or replace function  calculate_the_profit_of_order(id number) return number is
v_profit number;
cursor c1 is
 select sum(o.price - o.cost -(x.delay*0.001*(to_number(replace(p.list_price,'.',',') )) )) as final_profit
    from products p
    join orders o on p.product_id=o.product_id
    join max_delay_per_order x on x.order_id=o.order_id
    where o.order_id=id;
begin
    open c1;
    fetch c1 into v_profit;
    close c1;
    return v_profit;
end;
```

Η παραπάνω συνάρτηση παίρνει το order_id και επιστρέφει το profit κάθε παραγγελίας.Για παράδειγμα για order_id 1,2,32,41,19 έχουμε:

```
SET SERVEROUTPUT ON;
BEGIN
     DBMS_OUTPUT.PUT_LINE(calculate_the_profit_of_order(1));
     DBMS_OUTPUT.PUT_LINE(calculate_the_profit_of_order(2));
     DBMS_OUTPUT.PUT_LINE(calculate_the_profit_of_order(32));
     DBMS_OUTPUT.PUT_LINE(calculate_the_profit_of_order(41));
     DBMS_OUTPUT.PUT_LINE(calculate_the_profit_of_order(19));
END;
/
```

Script Output

Task completed in 0,232 seconds

```
33,71
550,42
-1538,10302
-5228,74029
-17,042


PL/SQL procedure successfully completed.
```

Στη συνέχεια υλοποιούμε την διεργασία find_profit_per_order η οποία καταχωρεί στον πίνακα profit_per_order το profit ανα παραγγελία(δύο διαφορετικές υλοποιήσεις γιατι η κάθε μια λειτουργούσε μόνο σε ένα απο τους δύο μας,οπότε για σιγουριά βάλαμε και τις δύο).

```
create or replace procedure find_profit_per_order as
begin

    delete from profit_per_order;

    insert into profit_per_order(order_id,final_profit)
    select o.order_id ,sum(o.price - o.cost -(x.delay*0.001*(to_number(p.list_price, 999999.99) ) ) ) as final_profit
    from products p
    join orders o on p.product_id=o.product_id
    join max_delay_per_order x on x.order_id=o.order_id
    group by o.order_id;

end;
```

```
create or replace procedure find_profit_per_order as
begin

    delete from profit_per_order;

    insert into profit_per_order(order_id,final_profit)
    select o.order_id ,sum(o.price - o.cost -(x.delay*0.001*(to_number(replace(p.list_price,'.',',') )) )) as final_profit
    from products p
    join orders o on p.product_id=o.product_id
    join max_delay_per_order x on x.order_id=o.order_id
    group by o.order_id;

end;

execute find_profit_per_order;
```

Η μοναδική διαφορά τους είναι στο select που χρησιμοποιούμε την to_number().

iii) Υπολογισμός κέρδους ανα παραγγελία και καταχώρηση σε πίνακες.

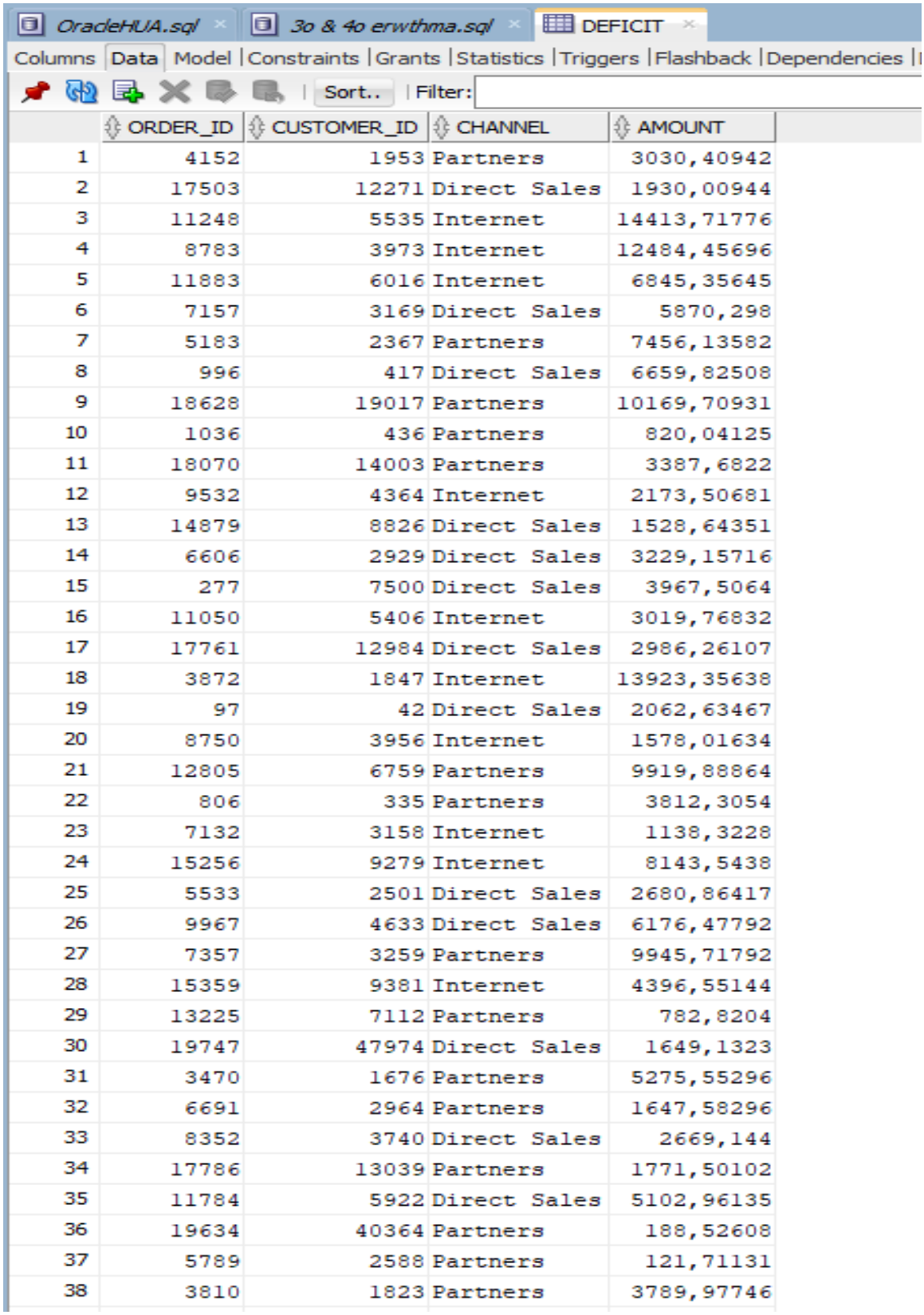
Δημιουργούμε τα tables deficit για τις ζημιές και profit για τα κέρδη και υλοποιούμε την διεργασία

FinalProfit η οποία διαφοροποιεί τις ζημιές και τα κέρδη και τα κάνει insert στον κατάλληλο πίνακα.Αν η διεργασια δεν τρέχει αλλάξτε την to_number και βάλτε (to_number(p.list_price, 999999.99)).

```
create table deficit(
order_id number ,
customer_id number ,
channel varchar2(20) ,
amount number
);

create table profit(
order_id number ,
customer_id number ,
channel varchar2(20) ,
amount number
);
```

```
CREATE OR REPLACE PROCEDURE FinalProfit as
v_order_id orders.order_id%type;
v_customer_id orders.customer_id%type;
v_channel orders.channel%type;
v_total_profit number;

CURSOR profit_cursor IS
    select o.order_id, o.customer_id, o.channel,
    sum(o.price - o.cost -(x.delay*0.001*(to_number(replace(p.list_price,'.',',') )) )) as final_profit
    FROM orders o
    JOIN products p ON p.product_id = o.product_id
    JOIN max_delay_per_order x ON x.order_id = o.order_id
    GROUP BY o.order_id, o.customer_id, o.channel;

BEGIN

    DELETE FROM deficit;
    DELETE FROM profit;


    FOR cursor_row IN profit_cursor LOOP
        v_order_id := cursor_row.order_id;
        v_customer_id := cursor_row.customer_id;
        v_channel := cursor_row.channel;
        v_total_profit := cursor_row.final_profit;

        IF v_total_profit < 0 THEN

            INSERT INTO deficit VALUES (v_order_id, v_customer_id, v_channel, ABS(v_total_profit));

        ELSIF v_total_profit > 0 THEN

            INSERT INTO profit VALUES (v_order_id, v_customer_id, v_channel, v_total_profit);

        END IF;
    END LOOP;


END;

execute finalprofit;
```

Ο πίνακας DEFICIT:



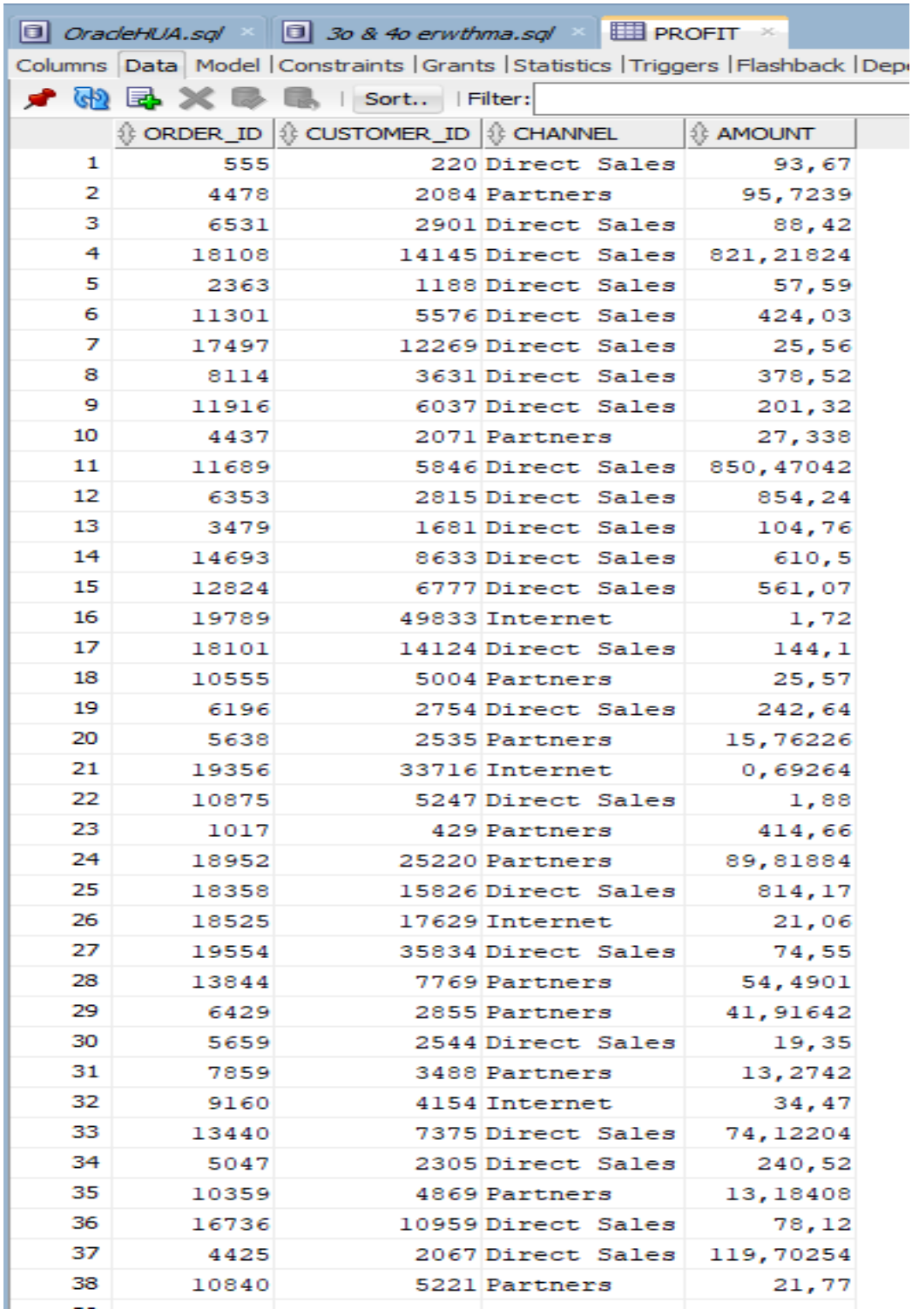
OracleHUA.sql | 3o & 4o erwthma.sql | DEFICIT

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies

Sort.. | Filter:

	ORDER_ID	CUSTOMER_ID	CHANNEL	AMOUNT
1	4152	1953	Partners	3030,40942
2	17503	12271	Direct Sales	1930,00944
3	11248	5535	Internet	14413,71776
4	8783	3973	Internet	12484,45696
5	11883	6016	Internet	6845,35645
6	7157	3169	Direct Sales	5870,298
7	5183	2367	Partners	7456,13582
8	996	417	Direct Sales	6659,82508
9	18628	19017	Partners	10169,70931
10	1036	436	Partners	820,04125
11	18070	14003	Partners	3387,6822
12	9532	4364	Internet	2173,50681
13	14879	8826	Direct Sales	1528,64351
14	6606	2929	Direct Sales	3229,15716
15	277	7500	Direct Sales	3967,5064
16	11050	5406	Internet	3019,76832
17	17761	12984	Direct Sales	2986,26107
18	3872	1847	Internet	13923,35638
19	97	42	Direct Sales	2062,63467
20	8750	3956	Internet	1578,01634
21	12805	6759	Partners	9919,88864
22	806	335	Partners	3812,3054
23	7132	3158	Internet	1138,3228
24	15256	9279	Internet	8143,5438
25	5533	2501	Direct Sales	2680,86417
26	9967	4633	Direct Sales	6176,47792
27	7357	3259	Partners	9945,71792
28	15359	9381	Internet	4396,55144
29	13225	7112	Partners	782,8204
30	19747	47974	Direct Sales	1649,1323
31	3470	1676	Partners	5275,55296
32	6691	2964	Partners	1647,58296
33	8352	3740	Direct Sales	2669,144
34	17786	13039	Partners	1771,50102
35	11784	5922	Direct Sales	5102,96135
36	19634	40364	Partners	188,52608
37	5789	2588	Partners	121,71131
38	3810	1823	Partners	3789,97746

Ο πίνακας PROFIT:



OracleHUA.sql | 3ο & 4ο erwthma.sql | PROFIT

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dep

Sort.. | Filter:

	ORDER_ID	CUSTOMER_ID	CHANNEL	AMOUNT
1	555	220	Direct Sales	93,67
2	4478	2084	Partners	95,7239
3	6531	2901	Direct Sales	88,42
4	18108	14145	Direct Sales	821,21824
5	2363	1188	Direct Sales	57,59
6	11301	5576	Direct Sales	424,03
7	17497	12269	Direct Sales	25,56
8	8114	3631	Direct Sales	378,52
9	11916	6037	Direct Sales	201,32
10	4437	2071	Partners	27,338
11	11689	5846	Direct Sales	850,47042
12	6353	2815	Direct Sales	854,24
13	3479	1681	Direct Sales	104,76
14	14693	8633	Direct Sales	610,5
15	12824	6777	Direct Sales	561,07
16	19789	49833	Internet	1,72
17	18101	14124	Direct Sales	144,1
18	10555	5004	Partners	25,57
19	6196	2754	Direct Sales	242,64
20	5638	2535	Partners	15,76226
21	19356	33716	Internet	0,69264
22	10875	5247	Direct Sales	1,88
23	1017	429	Partners	414,66
24	18952	25220	Partners	89,81884
25	18358	15826	Direct Sales	814,17
26	18525	17629	Internet	21,06
27	19554	35834	Direct Sales	74,55
28	13844	7769	Partners	54,4901
29	6429	2855	Partners	41,91642
30	5659	2544	Direct Sales	19,35
31	7859	3488	Partners	13,2742
32	9160	4154	Internet	34,47
33	13440	7375	Direct Sales	74,12204
34	5047	2305	Direct Sales	240,52
35	10359	4869	Partners	13,18408
36	16736	10959	Direct Sales	78,12
37	4425	2067	Direct Sales	119,70254
38	10840	5221	Partners	21,77

iv) Έσοδα και ζημίες από άντρες και γυναίκες.

```
select c.gender,sum(case when t.profit > 0 THEN t.profit ELSE 0 END) as esoda,sum(case when t.profit < 0 THEN t.profit ELSE 0 END) as zhmies
from customers c join(
select   distinct final_profit as profit,customer_id,p.order_id
from profit_per_order p join orders o on p.order_id=o.order_id)t on t.customer_id=c.customer_id
group by c.gender;
```

Τρέχοντας το παραπάνω query μας επιστρέφονται τα έσοδα και οι ζημίες από άνδρες και γυναίκες αντίστοιχα.

	GENDER	ESODA	ZHMIES
1	Male	1429050.19046	-33565349.893
2	Female	784236.05303	-18472039.40807

v) Έσοδα και ζημίες ανά κανάλι παραγγελιών.

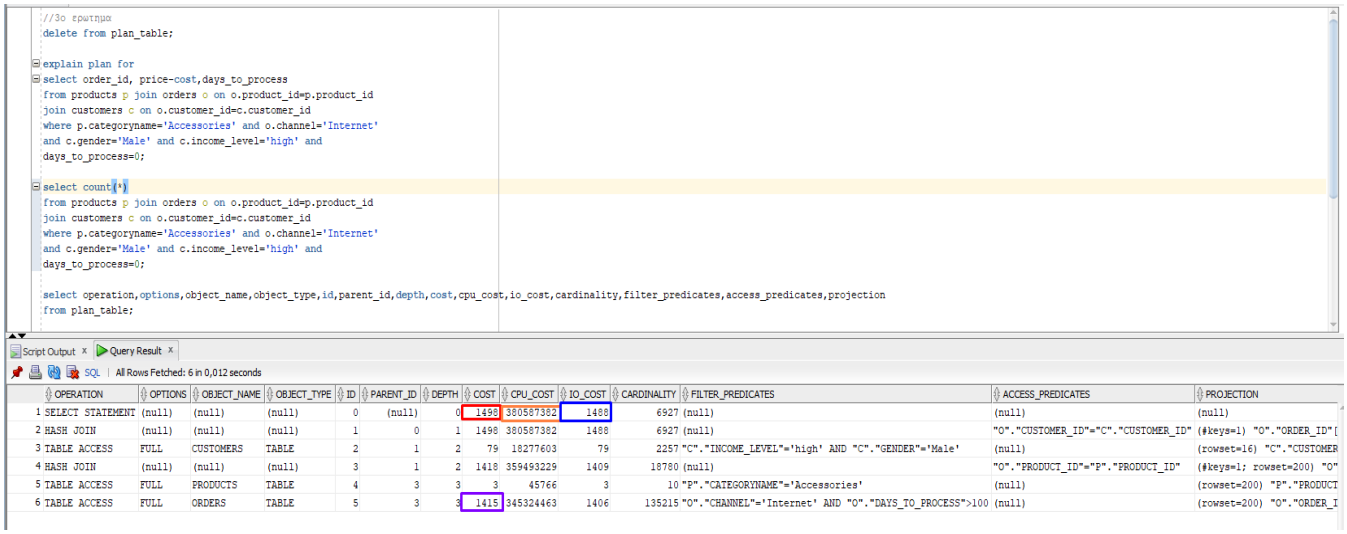
```
select t.channel,sum(case when t.profit > 0 THEN t.profit ELSE 0 END) as esoda,sum(case when t.profit < 0 THEN t.profit ELSE 0 END) as zhmies
from(
select  distinct final_profit as profit,p.order_id,o.order_id,o.channel as channel
from profit_per_order p join orders o on p.order_id=o.order_id) t
group by t.channel;
```

Τρέχοντας το παραπάνω query μας επιστρέφονται τα έσοδα και οι ζημίες ανά κανάλι παραγγελιών.

	CHANNEL	ESODA	ZHMIES
1	Partners	451621.92222	-14679080.81336
2	Direct Sales	1546636.26927	-26640436.65764
3	Internet	215028.052	-10717871.83007

Ερώτημα 3

1) Εκτιμώμενο συνολικό κόστος μέσω της EXPLAIN, Εύρεση CPU_COST , IO_COST και πιο χρονοβόρας ενέργειας.



```
//3ο ερωτημα
delete from plan_table;

explain plan for
select order_id, price-cost,days_to_process
from products p join orders o on o.product_id=p.product_id
join customers c on o.customer_id=c.customer_id
where p.categoryname='Accessories' and o.channel='Internet'
and c.gender='Male' and c.income_level='high' and
days_to_process=0;

select count(*)
from products p join orders o on o.product_id=p.product_id
join customers c on o.customer_id=c.customer_id
where p.categoryname='Accessories' and o.channel='Internet'
and c.gender='Male' and c.income_level='high' and
days_to_process=0;

select operation,options,object_name,object_type,id,parent_id,depth,cost,cpu_cost,io_cost,cardinality,filter_predicates,access_predicates,projection
from plan_table;
```

Script Output × Query Result ×

SQL | All Rows Fetched: 6 in 0,012 seconds

	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE	ID	PARENT_ID	DEPTH	COST	CPU_COST	IO_COST	CARDINALITY	FILTER_PREDICATES	ACCESS_PREDICATES	PROJECTION
1	SELECT STATEMENT	(null)	(null)	(null)	0	(null)	0	1498	380587382	1488	6927	(null)	(null)	(null)
2	HASH JOIN	(null)	(null)	(null)	1	0	1	1498	380587382	1488	6927	(null)	"O"."CUSTOMER_ID"="C"."CUSTOMER_ID"	(#keys=1) "O"."ORDER_ID"[
3	TABLE ACCESS	FULL	CUSTOMERS	TABLE	2	1	2	79	18277603	79	2257	"C"."INCOME_LEVEL"='high' AND "C"."GENDER"='Male'	(null)	(rowset=16) "C"."CUSTOMER
4	HASH JOIN	(null)	(null)	(null)	3	1	2	1418	359493229	1409	18780	(null)	"O"."PRODUCT_ID"="P"."PRODUCT_ID"	(#keys=1; rowset=200) "O"
5	TABLE ACCESS	FULL	PRODUCTS	TABLE	4	3	3	3	45766	3	10	"P"."CATEGORYNAME"='Accessories'	(null)	(rowset=200) "P"."PRODUCT
6	TABLE ACCESS	FULL	ORDERS	TABLE	5	3	3	1415	345324463	1406	135215	"O"."CHANNEL"='Internet' AND "O"."DAYS_TO_PROCESS">100	(null)	(rowset=200) "O"."ORDER_I

[Σύνδεσμος με την φωτογραφία σε καλύτερη ανάλυση]

Το συνολικό κόστος είναι **1497**, το CPU_COST είναι **380587382**, το IO_COST είναι **1488**. Η πιο χρονοβόρα ενέργεια είναι η TABLE_ACCESS στον πίνακα ORDERS η οποία έχει κόστος **1415.**

2) Εκτιμώμενο πλήθος αποτελεσμάτων για το ερώτημα, πλήθος πλειάδων που επιστρέφει, πλάνο εκτέλεσης του optimizer.

Script Output × Query Result ×

SQL | All Rows Fetched: 6 in 0,012 seconds

	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE	ID	PARENT_ID	DEPTH	COST	CPU_COST	IO_COST	CARDINALITY
1	SELECT STATEMENT	(null)	(null)	(null)	0	(null)	0	1498	380587382	1488	6927
2	HASH JOIN	(null)	(null)	(null)	1	0	1	1498	380587382	1488	6927
3	TABLE ACCESS	FULL	CUSTOMERS	TABLE	2	1	2	79	18277603	79	2257
4	HASH JOIN	(null)	(null)	(null)	3	1	2	1418	359493229	1409	18780
5	TABLE ACCESS	FULL	PRODUCTS	TABLE	4	3	3	3	45766	3	10
6	TABLE ACCESS	FULL	ORDERS	TABLE	5	3	3	1415	345324463	1406	135215

Το εκτιμώμενο πλήθος αποτελεσμάτων,δηλαδή το cardinality είναι **6927**.Το πραγματικό είναι **92.**

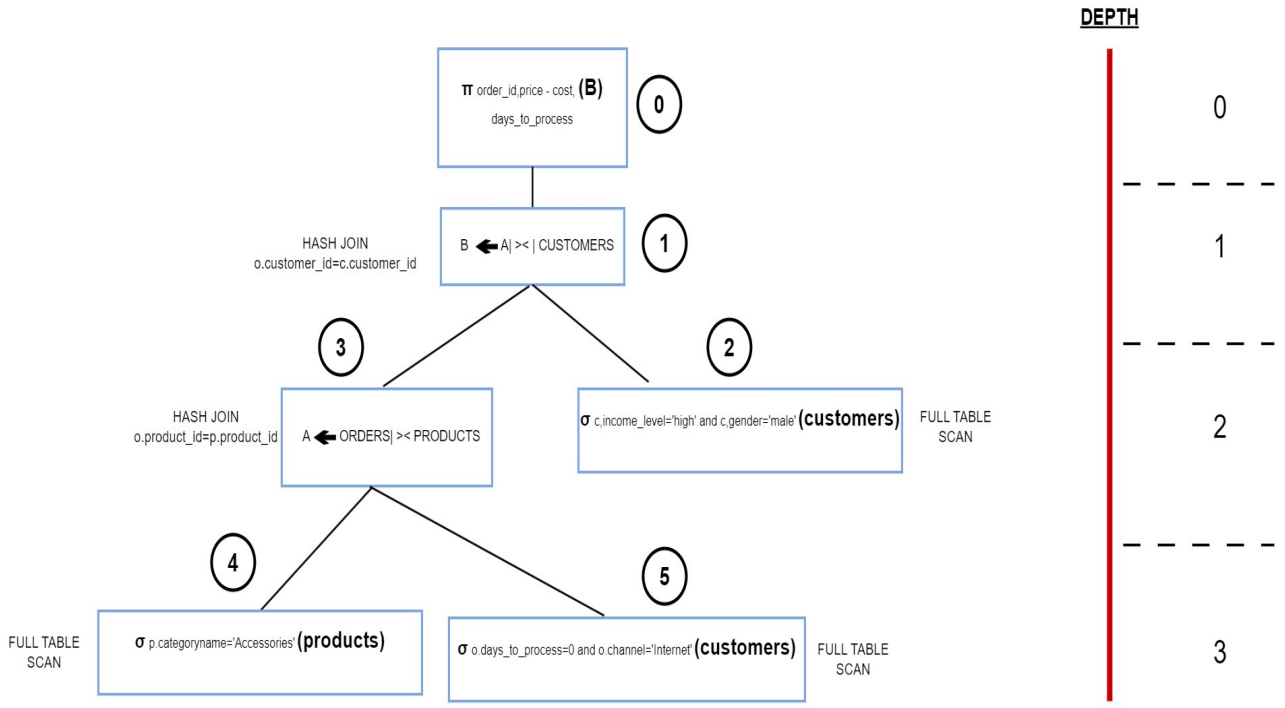
```
select count(*)
from products p join orders o on o.product_id=p.product_id
join customers c on o.customer_id=c.customer_id
where p.categoryname='Accessories' and o.channel='Internet'
and c.gender='Male' and c.income_level='high' and
days_to_process=0;
```

Script Output × Query Result ×

SQL | All Rows Fetched: 1 in 0,078 seconds

	COUNT(*)
1	92

Το πλάνο εκτέλεσης που επέλεξε ο optimizer:



3) Τελικό κόστος μετά τη βελτιστοποίηση της σχεδίασης.

Δημιουργώντας τα παρακάτω ευρετήρια:

```
CREATE INDEX categoryname_idx ON products(categoryname);
CREATE INDEX p_orders_idx ON orders(product_id);
CREATE  bitmap INDEX o_channel_idx ON orders(channel);
CREATE INDEX c_id_idx ON customers(customer_id);
CREATE  bitmap INDEX c_gender_idx ON customers(gender);
CREATE  bitmap INDEX c_income_level_idx_ ON customers(income_level);
CREATE INDEX days_to_process_idx ON orders(days_to_process);
```

Και τρέχοντας την explain για το ερώτημα:

```
delete from plan_table;

explain plan for
select order_id, price-cost,days_to_process
from products p join orders o on o.product_id=p.product_id
join customers c on o.customer_id=c.customer_id
where p.categoryname='Accessories' and o.channel='Internet'
and c.gender='Male' and c.income_level='high' and
days_to_process=0;
```

Script Output | Query Result

SQL | All Rows Fetched: 27 in 0,014 seconds

	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE	ID	PARENT_ID	DEPTH	COST	CPU_COST	IO_COST
1	SELECT STATEMENT	(null)	(null)	(null)	0	(null)	0	1265	46510866	1264
2	HASH JOIN	(null)	(null)	(null)	1	0	1	1265	46510866	1264
3	TABLE ACCESS	BY INDEX ROWID BATCHED	CUSTOMERS	TABLE	2	1	2	77	1433899	77
4	BITMAP CONVERSION	TO ROWIDS	(null)	(null)	3	2	3	(null)	(null)	(null)
5	BITMAP AND	(null)	(null)	(null)	4	3	4	(null)	(null)	(null)
6	BITMAP INDEX	SINGLE VALUE	C_INCOME_LEVEL_IDX_	INDEX (BITMAP)	5	4	5	(null)	(null)	(null)
7	BITMAP INDEX	SINGLE VALUE	C_GENDER_IDX	INDEX (BITMAP)	6	4	5	(null)	(null)	(null)
8	HASH JOIN	(null)	(null)	(null)	7	1	2	1188	43932917	1187
9	NESTED LOOPS	(null)	(null)	(null)	8	7	3	1188	43932917	1187
10	NESTED LOOPS	(null)	(null)	(null)	9	8	4	(null)	(null)	(null)
11	STATISTICS COLLECTOR	(null)	(null)	(null)	10	9	5	(null)	(null)	(null)
12	TABLE ACCESS	BY INDEX ROWID BATCHED	PRODUCTS	TABLE	11	10	6	2	18143	2
13	INDEX	RANGE SCAN	CATEGORYNAME_IDX	INDEX	12	11	7	1	9121	1
14	BITMAP CONVERSION	TO ROWIDS	(null)	(null)	13	9	5	(null)	(null)	(null)
15	BITMAP AND	(null)	(null)	(null)	14	13	6	(null)	(null)	(null)
16	BITMAP INDEX	SINGLE VALUE	O_CHANNEL_IDX	INDEX (BITMAP)	15	14	7	(null)	(null)	(null)
17	BITMAP CONVERSION	FROM ROWIDS	(null)	(null)	16	14	7	(null)	(null)	(null)
18	INDEX	RANGE SCAN	P_ORDERS_IDX	INDEX	17	16	8	26	2730607	26
19	BITMAP CONVERSION	FROM ROWIDS	(null)	(null)	18	14	7	(null)	(null)	(null)
20	INDEX	RANGE SCAN	DAYS_TO_PROCESS_IDX	INDEX	19	18	8	174	18493781	174
21	TABLE ACCESS	BY INDEX ROWID	ORDERS	TABLE	20	8	4	1186	41833674	1185
22	TABLE ACCESS	BY INDEX ROWID BATCHED	ORDERS	TABLE	21	7	3	1186	41833674	1185
23	BITMAP CONVERSION	TO ROWIDS	(null)	(null)	22	21	4	(null)	(null)	(null)
24	BITMAP AND	(null)	(null)	(null)	23	22	5	(null)	(null)	(null)
25	BITMAP INDEX	SINGLE VALUE	O_CHANNEL_IDX	INDEX (BITMAP)	24	23	6	(null)	(null)	(null)
26	BITMAP CONVERSION	FROM ROWIDS	(null)	(null)	25	23	6	(null)	(null)	(null)
27	INDEX	RANGE SCAN	DAYS_TO_PROCESS_IDX	INDEX	26	25	7	175	18500052	175

Βλέπουμε πως το τελικό κόστος έφτασε το **1265.**

Ερώτημα 4

1) Εκτιμώμενο συνολικό κόστος μέσω της EXPLAIN, Εύρεση CPU_COST , IO_COST και πιο χρονοβόρας ενέργειας.

```
//4ο ερωτημα

delete from plan_table;

explain plan for
select order_id, price-cost,days_to_process
from products p join orders o on o.product_id=p.product_id
join customers c on o.customer_id=c.customer_id
where p.categoryname='Accessories' and o.channel='Internet'
and c.gender='Male' and c.income_level='high' and
days_to_process>100;

select operation,options,object_name,object_type,id,parent_id,depth,cost,cpu_cost,io_cost,cardinality,filter_predicates,access_predicates,projection
from plan_table;
```

Script Output | Query Result

All Rows Fetched: 6 in 0,018 seconds

	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE	ID	PARENT_ID	DEPTH	COST	CPU_COST	IO_COST	CARDINALITY	FILTER_PREDICATES	ACCESS_PREDICATES
1	SELECT STATEMENT	(null)	(null)	(null)	0	(null)	0	1498	395682082	1488	13541	(null)	(null)
2	HASH JOIN	(null)	(null)	(null)	1	0	1	1498	395682082	1488	13541	(null)	"O"."CUSTOMER_ID"="C"."CUSTOMER_ID"
3	TABLE ACCESS	FULL	CUSTOMERS	TABLE	2	1	2	79	18277603	79	6938	"C"."INCOME_LEVEL"='high' AND "C"."GENDER"='Male'	(null)
4	HASH JOIN	(null)	(null)	(null)	3	1	2	1418	374409579	1409	13542	(null)	"O"."PRODUCT_ID"="P"."PRODUCT_ID"
5	TABLE ACCESS	FULL	PRODUCTS	TABLE	4	3	3	3	45766	3	3	"P"."CATEGORYNAME"='Accessories'	(null)
6	TABLE ACCESS	FULL	ORDERS	TABLE	5	3	3	1415	345324463	1406	284389	"O"."CHANNEL"='Internet' AND "O"."DAYS_TO_PROCESS">100	(null)

Σύνδεσμος με την φωτογραφία σε καλύτερη ανάλυση

Το συνολικό κόστος είναι **1498**, το CPU_COST είναι **395682082**, το IO_COST είναι **1488**. Η πιο χρονοβόρα ενέργεια είναι η TABLE_ACCESS στον πίνακα ORDERS η οποία έχει κόστος **1415.**

2) Τελικό κόστος μετά τη βελτιστοποίηση της σχεδίασης.

Δημιουργώντας τα παρακάτω ευρετήρια:

```
create  index customer_idx on customers(customer_id,gender,income_level);
create index order_idx on orders(product_id,days_to_process);
create  bitmap index o_channel_idx ON orders(channel);
create index product_idx on products(categoryname);
```

Και τρέχοντας την explain για το ερώτημα:

```
//4ο ερώτημα

delete from plan_table;

explain plan for
select order_id, price-cost,days_to_process
from products p join orders o on o.product_id=p.product_id
join customers c on o.customer_id=c.customer_id
where p.categoryname='Accessories' and o.channel='Internet'
and c.gender='Male' and c.income_level='high' and
days_to_process>100;
```

Script Output | Query Result

SQL | All Rows Fetched: 17 in 0,009 seconds

	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_TYPE	ID	PARENT_ID	DEPTH	COST	CPU_COST	IO_COST
1	SELECT STATEMENT	(null)	(null)	(null)	0	(null)	0	1475	354704056	1466
2	HASH JOIN	(null)	(null)	(null)	1	0	1	1475	354704056	1466
3	INDEX	FAST FULL SCAN	CUSTOMER_IDX	INDEX	2	1	2	58	11082917	58
4	HASH JOIN	(null)	(null)	(null)	3	1	2	1417	341936089	1408
5	NESTED LOOPS	(null)	(null)	(null)	4	3	3	1417	341936089	1408
6	NESTED LOOPS	(null)	(null)	(null)	5	4	4	(null)	(null)	(null)
7	STATISTICS COLLECTOR	(null)	(null)	(null)	6	5	5	(null)	(null)	(null)
8	TABLE ACCESS	BY INDEX ROWID BATCHED	PRODUCTS	TABLE	7	6	6	2	18143	2
9	INDEX	RANGE SCAN	PRODUCT_IDX	INDEX	8	7	7	1	9121	1
10	BITMAP CONVERSION	TO ROWIDS	(null)	(null)	9	5	5	(null)	(null)	(null)
11	BITMAP AND	(null)	(null)	(null)	10	9	6	(null)	(null)	(null)
12	BITMAP INDEX	SINGLE VALUE	O_CHANNEL_IDX	INDEX (BITMAP)	11	10	7	(null)	(null)	(null)
13	BITMAP CONVERSION	FROM ROWIDS	(null)	(null)	12	10	7	(null)	(null)	(null)
14	SORT	ORDER BY	(null)	(null)	13	12	8	(null)	(null)	(null)
15	INDEX	RANGE SCAN	ORDER_IDX	INDEX	14	13	9	16	1276993	16
16	TABLE ACCESS	BY INDEX ROWID	ORDERS	TABLE	15	4	4	1414	335941446	1406
17	TABLE ACCESS	FULL	ORDERS	TABLE	16	3	3	1414	335941446	1406

Παρατηρούμε ότι το τελικό κόστος φτάνει στο **1475.**

Sources :

1ο ερώτημα:

Βίβλιο “Θεμελιώδεις Αρχές Συστημάτων Βάσεων Δεδομένων 7η “Έκδοση”

κεφάλαιο 17.3 σελιδες 510-524

1ο έως 4ο ερώτημα:

Διαφάνειες και υλικό μαθήματος στο eclass.