```
create table AGENTI (
A_IDAgente int,
A_Nome varchar2(20),
A_Cogome varchar2(20),
A_DataN date,
primary key (A_IDAgente)
);
create table PRODOTTI (
P_IDProdotto int,
P_Nome varchar2(20),
P_Categoria varchar2(20),
P_PrezzoUnitario int,
primary key (P_IDProdotto)
);
create table VENDITA (
V_IDProdotto int,
V_IDCliente int,
V_Data date,
V_Quantita int,
V_PrezzoTotale int,
V_IDAgente int,
primary key (V_IDProdotto, V_IDCliente, V_Data),
foreign key (V_IDProdotto) references PRODOTTI (P_IDProdotto),
foreign key (V_IDAgente) references AGENTI (A_IDAgente)
);
create table PERCENTUALI (
PE_QuantitaMin int,
PE_Percentuale int,
primary key (PE_Quantita),
);
create table PROFITTI (
PR_IDAgente int,
PR_Data date,
PR_Profitto int,
primary key (PR_IDAgente, PR_Data),
foreign key (PR_IDAgente) references AGENTI (A_IDAgente)
);
```

```
create or replace
procedure CalcolaProfitti(vDataFin date) IS
--cursore
cursor cVendite(DataInit date) is
select V_IDAgente, SUM(V_Quantita) AS Quantita, SUM(V_PrezzoTotale) AS Totale
FROM VENDITA
WHERE V_Data> DataInit AND V_Data< vDataFin
GROUP BY V_IDAgente;

vVendite cVendite%ROWTYPE;
vDataInit date;
vProfitto float;
vPercentuale float;
begin
--calcolo data inizio periodo
SELECT MAX(PR_Data) into vDataInit FROM PROFITTI;

vProfitto:=0;
open cVendite(vDataInit);
loop
        fetch cVendite into vVendite;
        exit when cVendite%NOTFOUND;
        --calcolo profitto
        --calcola percentuale
        SELECT MAX(PE_Percentuale) INTO vPercentuale
        FROM PERCENTUALI
        WHERE PE_QuantitaMin<vVendite.Quantita;
        --aggiorna Profitto
        vProfitto:= vPercentuale*vVendite.Totale;

        INSERT INTO PROFITTI VALUES (vVendite.V_IDAgente, vDataFin, vProfitto);
end loop;
close cVendite;
end;
```

Dati da statistiche:
NT_O= 1.500.000 Len(O)=109
NT_C= 150.000 Len(C)=158

$NP_O = \lceil 1.500.000 \times 109 / (4.096 \times 0{,}69) \rceil = 57.851$

$NP_C = \lceil 150.000 \times 158 / (4.096 \times 0{,}69) \rceil = 8{,}386$
Sel(C_MKTSEGMENT='AUTOMOBILE') = 1/5

Si accede all'indice su O_CUSTKEY (IX_CUST_ORDERS) in base al valore di un customers al fine di recuperare gli ordini relativi a un cliente. L'operazione è ripetuta per tutti i clienti selezionati dal predicato su C_MKTSEGMENT='AUTOMOBILE'

$NL_{O\_CUSTKEY} = \lceil (1.500.000 \times 4 + 150.000 \times 4) / (4.096 \times 0{,}69) \rceil = 2.336$
Costo di accesso mediante l'indice su PS_PARTKEY:
$2 + \lceil 1/150.000 \times 2.336 \rceil + \Phi(150.000/1.500.000, 57.851) = 2+1+10 = 13$

Costo(Nested Loop C-O) = $8.386 + \lceil 1/5 \times 150.000 \rceil \times 13 =$ **388.386**

Il numero di tuple in risultato al join è pari alla cardinalità di ORDERS a cui è applicato il filtro di selezione

NT_{C+O} = $NT_O \times$ Sel(C_MKTSEGMENT='AUTOMOBILE') = $\lceil 1.500.000 / 5 \rceil = 300.000$
$NP_{C+O} = \lceil 300.000 \times (158+109) / (4096 \times 0{,}69) \rceil = 28.342$

Costo (GB)=$2 \times 28.342 \times (\lceil \log_{100}(28.342) \rceil + 1) = 2 \times 28.342 \times 4 =$ **226.736**

Per il calcolo delle tuple attese dopo il GROUP BY si dovrebbe utilizzare Cardenas ma considerando che la cardinalità di C_NATIONKEY è molto limitata il risultato è scontato

NT_{GB} = NK(C_NATIONKEY) = 25
$NP_{GB} = \lceil 25 \times (4+4) / (4.096 \times 0{,}69) \rceil = 1$

L'ordinamento per la clausila ORDER BY si può fare in memoria poiché il numero di pagine di NP_{GB} è minore del numero di buffer NB a disposizione, quindi non operando in pipeline, il costo di ordinamento è pari al costo di lettura della relazione

Costo totale = **388.386 + 226.736 + 1 = 615.123**