```
CREATE OR REPLACE PROCEDURE ricerca(vDataDa date, vBudgetPersonaGG numeric,
vTipologiaVacanza varchar2, vcittadestinazione varchar2)IS

 rilevanza numeric(5,2);
 r_distanza numeric (5,2);
 regione_citta citta.c_regione%TYPE;

CURSOR c_rilevanza IS
 SELECT c_citta, c_regione, 1-abs(di_datada- vDataDa)*0.1 AS scostamento, di_id,
  vbudgetpersonagg/di_prezzopersonagg as budgetrelativo
 FROM destinazione, disponibilita, citta
 WHERE de_id = di_struttura AND de_citta = c_citta
 AND de_tipologiavacanza = vTipologiaVacanza
 AND di_datada between vDataDa-9 AND vDataDa + 9
 AND vbudgetpersonagg/di_prezzopersonagg>=0.3;

BEGIN
--Ottengo la regione in cui si trova la citta richiesta
 SELECT c_regione INTO regione_citta FROM citta
 WHERE c_citta = vcittadestinazione;

FOR ril in c_rilevanza LOOP

   --Imposto la rilevanza della citta
   r_distanza :=0.2;
   IF vcittadestinazione = ril.c_citta THEN
     r_distanza :=1;
   ELSIF regione_citta = ril.c_regione THEN
     r_distanza :=0.6
   END IF;

   rilevanza:= (r_distanza * ril.scostamento * ril.budgetrelativo);

   dbms_output.put_line('La disponibilita con id '|| v_ril.di_id ||' ha
   rilevanza uguale a '|| rilevanza);
END LOOP;
END;
```

Le principali categorie di errori commessi sono stati:

1) Non effettuare i calcoli su data e budget nel cursore
2) Non effettuare i controlli di > 0 nel cursore. Questo ha comportato il recupero di tuple escludibili a priori
3) Eseguire la query SQL per recuperare la regione all'interno del ciclo: la regione è legata alla città passata come parametro e quindi il risultato non dipende dalla disponibilità in corso di verifica

```
explain plan for
      SELECT L_RETURNFLAG,COUNT(*)
      FROM LINEITEM,PART
      WHERE L_PARTKEY=P_PARTKEY AND P_TYPE='STANDARD PLATED TIN' AND L_DISCOUNT
      <=0.05
      group by L_RETURNFLAG;
      @?/RDBMS/ADMIN/UTLXPLS;

Plan Table
--------------------------------------------------------------------------------
| Operation                 |  Name    |  Rows | Bytes|  Cost  | Pstart| Pstop |
--------------------------------------------------------------------------------
| SELECT STATEMENT          |          |     3 |  111 |  44791 |       |       |
|  SORT GROUP BY            |          |     3 |  111 |  44791 |       |       |
|   NESTED LOOPS            |          |    23K|  854K|  43850 |       |       |
|    TABLE ACCESS FULL      |PART      |     1K|   35K|   1162 |       |       |
|    TABLE ACCESS BY INDEX R|LINEITEM  |     3M|   33M|     32 |       |       |
|     INDEX RANGE SCAN      |IX_PART_L |     3M|      |      2 |       |       |
```

$NP_{LI} = \lceil 6.001.215 \times 113/ (4096 \times 0{,}69) \rceil = 239.944$
$NP_{P} = \lceil 200.000 \times 133/ (4096 \times 0{,}69) \rceil = 9.412$
Sarà accettata anche la lunghezza 130 per PART

Sel(`L_DISCOUNT <=0.05`) 0,05-0/0,1=50%
Sel(`P_TYPE='STANDARD PLATED TIN'`)= 1/150
$NL_{L\_PARTKEY} = \lceil (200.000 \times 4 + 6.001.215 \times 4) / (4.096 \times 0{,}69) \rceil = 8.777$

Costo di accesso all'indice $1 + \lceil 1/200.000 \times 8.777 \rceil + 1 \times \Phi(6.001.215/200.000, 239.944) = 1+1+31 = 33$

Costo(Nested $Loop_{LI\text{-}P}$) = **9.412 + 200.000** / **150** × **33= 53.412**

$NT_{LI+P} = \lceil 6.001.215 \times 0{,}5 \times 1/150 \rceil = 20.005$
$NP_{LI+P} = \lceil 20.005 \times (133+113)/ (4096 \times 0{,}69) \rceil = 1.742$

Costo $(GB_{LI+P}) = 2 \times 1.742 \times (\lceil \log_{100}(1.742) \rceil + 1) = 2 \times 1.742 \times (2+1) =$ **10.452**

Costo totale = 53.412 + 10.452 = 63.864