Question 1:- Perform heuristic optimization on query to

Find the Last names of employees born after 1957 who work on a project named ‘Audit’.

Answer:-

```
SELECT Last Name
FROM Employees
WHERE Year Of Birth >1957 AND EmployeeID IN (
        SELECT Employee ID
        FROM EmployeeProjects
        WHERE Project Name = 'Audit');
```

Using Heuristic approach -

```
SELECT E.LastName
FROM Employees E
INNER JOIN Employee Projects EP ON E.Employee ID=EP.EmployeeID
WHERE E.YearOfBirth > 1957 AND EP.ProjectName="Audit”;
```

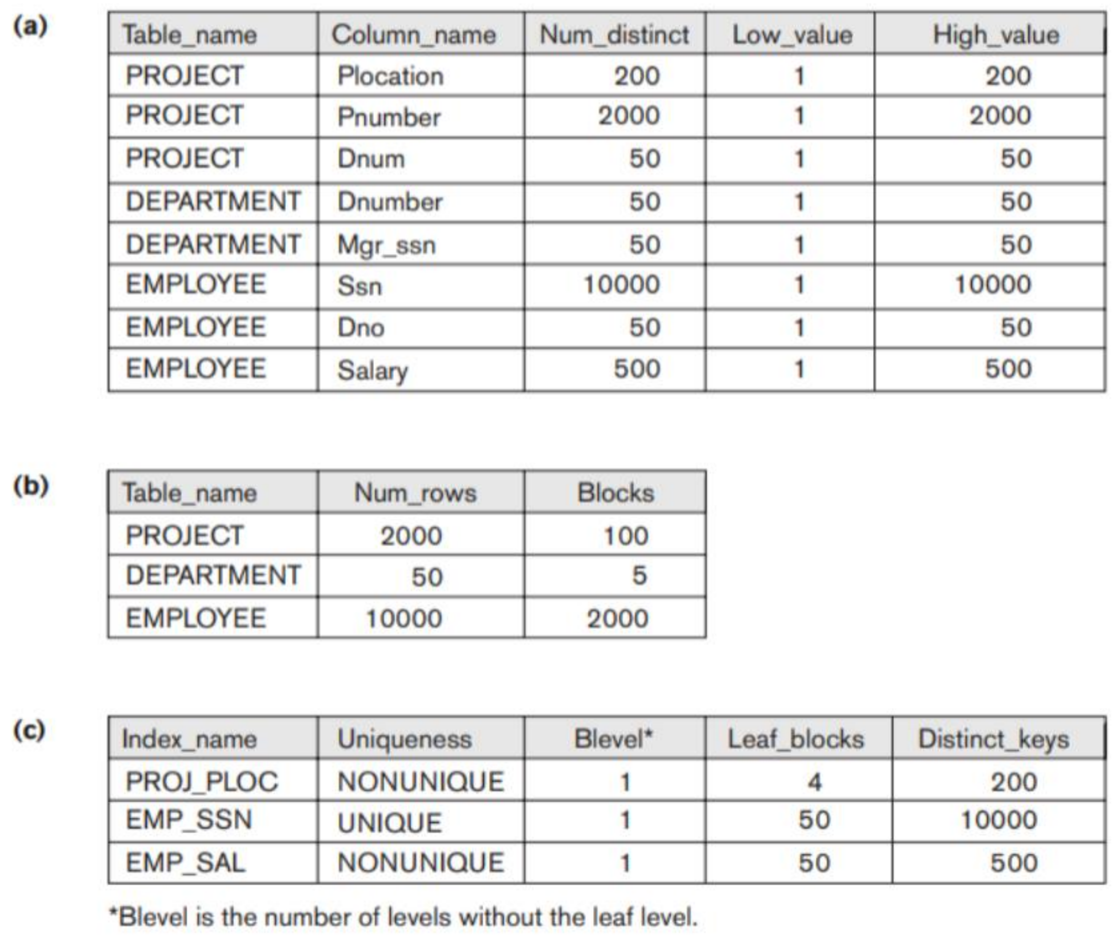
Question 2:- Illustrate cost based query optimization for query

SELECT PNUMBER, DNUM, LNAME, ADDRESS, BDATE

FROM PROJECT, DEPARTMENT, EMPLOYEE

WHERE DNUM = DNUMBER AND MGRSSN = SSN AND PLOCATION = “Chandigarh”;

Sample statistical information for relations in Q2. (a) Column information. (b) Table information. (c) Index information.



(a)

Table_name	Column_name	Num_distinct	Low_value	High_value
PROJECT	Plocation	200	1	200
PROJECT	Pnumber	2000	1	2000
PROJECT	Dnum	50	1	50
DEPARTMENT	Dnumber	50	1	50
DEPARTMENT	Mgr_ssn	50	1	50
EMPLOYEE	Ssn	10000	1	10000
EMPLOYEE	Dno	50	1	50
EMPLOYEE	Salary	500	1	500

(b)

Table_name	Num_rows	Blocks
PROJECT	2000	100
DEPARTMENT	50	5
EMPLOYEE	10000	2000

(c)

Index_name	Uniqueness	Blevel*	Leaf_blocks	Distinct_keys
PROJ_PLOC	NONUNIQUE	1	4	200
EMP_SSN	UNIQUE	1	50	10000
EMP_SAL	NONUNIQUE	1	50	500

*Blevel is the number of levels without the leaf level.

Answer:-

Cost Based Query optimization involves the DBMS estimating the cost of executing different query execution plans and choosing the most efficient plan based on these estimates.

Step by step approach:

1) Parsing and Initial Query Plan - The DBMS first parse the SQL query and generates an initial query plan.

2) Table access Order Optimization - The optimizer evaluates different table orders & join strategies based on statistics and metadata about the tables involved.

3) Filter Pushdown Optimizations - In this query filtering based on PLOCATION = 'Chandigarh’ can potentially be pushed on to the ‘PROJECT’ table.

4) Cost Estimation – The DBMS estimates the cost of executing different query plans based on factors like table sizes, indexes.

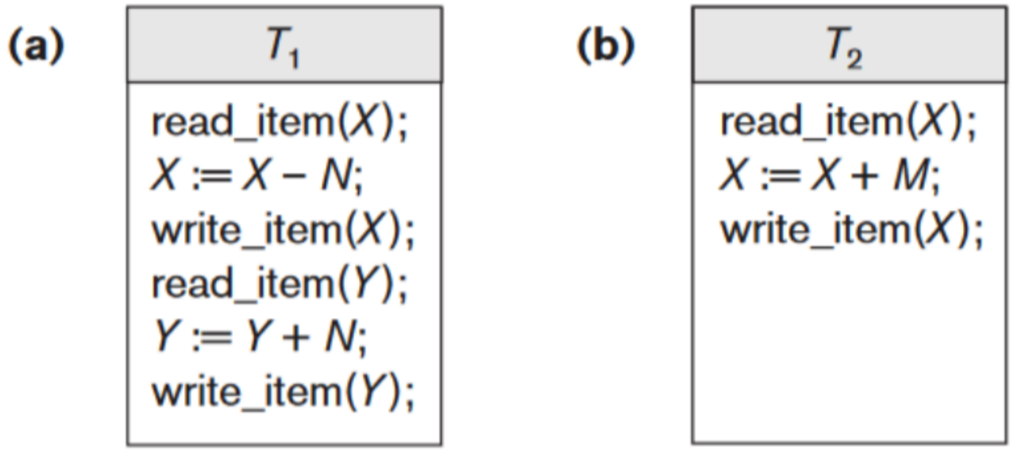
Question 3:- Why is an explicit transaction end statement needed in SQL but not an explicit begin statement?

Answer:-

In SQL, the need for an explicit 'BEGIN' statement versus an explicit 'COMMIT' or ‘ROLLBACK’ statement is related to the concept of transactions and the structure of SQL transaction.

Explicit 'COMMIT' and 'ROLLBACK' statement are essential for controlling & finalizing transaction, ensuring data integrity and handling errors effectively within the SQL database system.

Question 4:-



(a)

T_1
read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);

(b)

T_2
read_item(X);
$X := X + M$;
write_item(X);

Add the operation commit at the end of each of the transactions T1 and T2, and then list all possible schedules for the modified transactions. Determine which of the schedules are recoverable, which are cascade-less, and which are strict

Answer:-

Transaction after commit:

T_1: $R_1(X)$; $X=X-N$; $W_1(x)$; $R_1(Y)$; $Y=Y+N$; $W_1(Y)$; Commit;

T_2: $R_2(X)$; $X=X+M$; $W_2(x)$; Commit;

Possible Schedules

1) T_1: R(X); T_1: W(X); T_1: R(Y); T_1: W(Y); T_1: C;
   T_2: R(X); T_2: W(X); T_2: C;

2) T_1: R(X); T_1: W(X); T_1: C;
   T_2: R(X); T_2: W(X); T_2: C;
   T_1: R(Y); T_1: W(Y); T_1: C;

3) T_1: R(X); T_1: W(X); T_1: C;
   T_2: R(X); T_2: W(X); T_2: C;
   T_1: R(Y); T_1: W(Y); T_1: C;

4) T_1: R(X); T_1: W(X); T_1: C;
   T_2: R(X); T_2: W(X); T_2: C;
   T_1: R(Y); T_1: W(Y); T_1: C;

Recoverable Schedules –

- S3 and S4 are recoverable, since, in each of them T_1 commits only after all its reads have been written by T_1 itself or T2.

Cascade-less Schedules (Strict Schedules) –

- S1 and S3 are cascade-less, since, in each of T_1 commits only after its writes have been read by T_1 and T_2

- S2 is neither recoverable nor cascade-less.