①	Materialization	Pipeling
i)	It is a traditional approach to evaluate multiple operations.	i) It is a modern approach to evaluate multiple operations.
ii)	It uses temporary relations for storing the results of the evaluated operations. So it needs more temporary files and I/O.	ii) It does not use any temporary relations for storing the results of the evaluated operations.
iii)	It is less efficient as it takes time to generate the query results.	iv) It is more efficient way of query evaluation as it quickly generates the results.
iv)	It does not have any higher requirements for memory buffers for query evaluation.	iv) It requires memory buffer at a high rate for generating outputs. Insufficient memory buffer will cause thrashing.
v)	No thrashing occurs in materialization. Thus in cases, materialization is having better performance.	v) Poor performance if thrashing occurs.
vi)	The overall cost includes the cost of operations plus the cost of reading and writing results on temporary storage.	vi) It optimizes the cost of query evaluation. As it does not include the cost of reading and writing the temporary storages.

② Advantages & i) Reduces the search space for optimization strategy.

ii) Allows the query optimizes to be based on dynamic processing technique.
iii) Convinient for Pipelining.

Disadvantages: i) Reduction of search space might skip some low cost evaluation techniques.
ii) Only fully pipelined strategies are used.
iii) Many alternative execution strategies are not considered.

3) a) S1: $\sigma_{roomNo=1 \wedge hotelNo='H001'}(Room)$
SC = 1 as room no & Hotel no as key attribute of room using hash = Cost - 1
Linear search cost = $\lceil 10000/2000 \rceil /2 = 25$

S2: $\sigma_{type='D'}(Room)$
$SC_{type}(Room) = nTuples(Room)/nDistinct\ type(Room)$
$= 10000/10 = 1000$
Cost = $2+1000/200 = 5$
Linear search cost = 50

S3: $\sigma_{hotelNo='H002'}(Room)$
$SC_{hotelNo}(Room) = 10000/50 = 200$
Cost = $2+200/200 = 3$

S4: $\sigma_{price>100}(Room)$
Cost $2 + (50/2) + 1000/2 = 5027$
Linear search = 50

S5: $\sigma_{type='S' \wedge hotelNo='H003'}(Room)$

Secondary index searches costs are 5 & 3
Linear cost = 50

S6: $\sigma_{type = 'S' \vee price < 100}(Room)$
Use secondary index on type & price take union
linear search cost = 50

b) J1: (Hotel) ⋈ HotelNo (Room)
Assume NBuffer = 100
Block nested loop = nBlocks (Room) = 10000/200 = 50
nBlocks (Hotel) = 50/40 = 1
R = Hotel = S = Room
If buffer has 1 block for R & S
Cost = nBlock (R) + nBlock (R) & nBlocks (S)
= 1 + (50 × 2) = 102
If R (Hotel) fits no buffer
Cost = nBlock (R) + nBlocks (S) = 50 + 2 = 52
As Hotel is primary key
n blocks (R) + nTuples (R) × n Levels HotelNo + $\left(\frac{SC_A \cdot R}{Bfactor_R}\right)$
2 + 50 × (2 + 1) = 2 + 150 = 152
Sort merge
n blocks (R) * [$\log_2$ (nblocks (R))] + nBlocks (S) [$\log_2$ nblocks (S)]
= 2 * $\log_2$ (2) + 50 * [$\log_2$ (50)]
= 2 + 250 = 252
for merge = 50 + 2
Hash
If hash fits into memory
3 (nBlocks (R) + nBlocks (S)

J6 : (Booking) ⋈ roomNo (Room)

nBlocks (Booking) = 1667
nBlocks (Room) = 50
If 1 block is allowed =
· nBlock (R) → (nBlock(R) * nBlock (S))
= 50 + 50 * 1667 = 83400
If room fits in memory:
nBlocks (R) + nBlocks (S) = 50 + 1667 = 1717
Sort Merge
If unsorted nBlock (R) $\lceil \log_2 (n\text{Block}R) \rceil$ + nBlock(S)
= $50 \log_2 50$ + 1667 + $\log$ 1667 = 18381
Hash
3(1667+50) = 5151

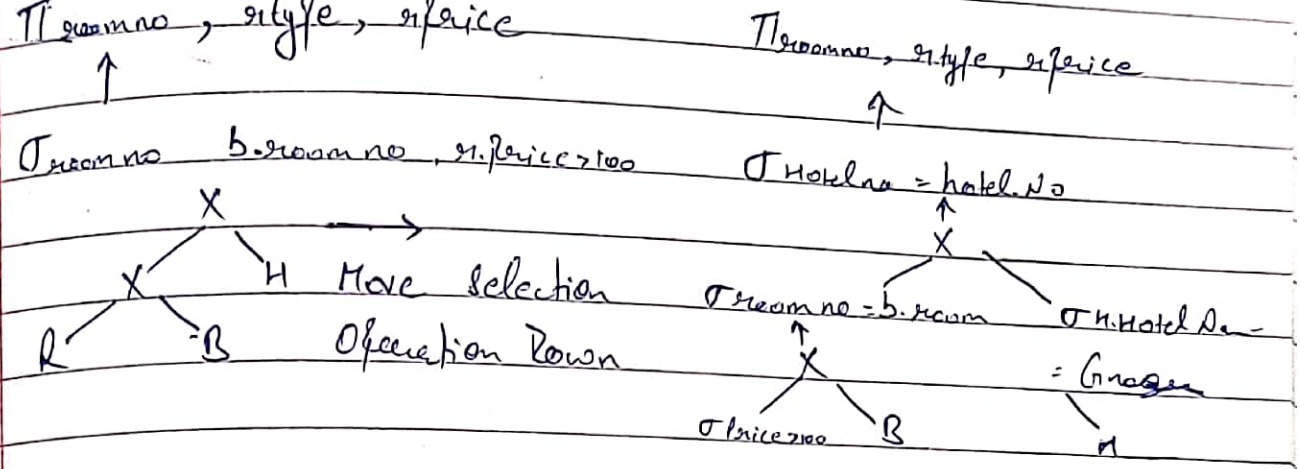
c) P1: Π_{hotelNo} (Hotel)
Using Sorting
nBlocks(R) + nBlocks(R) * $\log_2$ (nBlockR)
2 + 2 x $\log_2$ 2 = 4
Using hashing
As hotel no is Pk, cost remains same

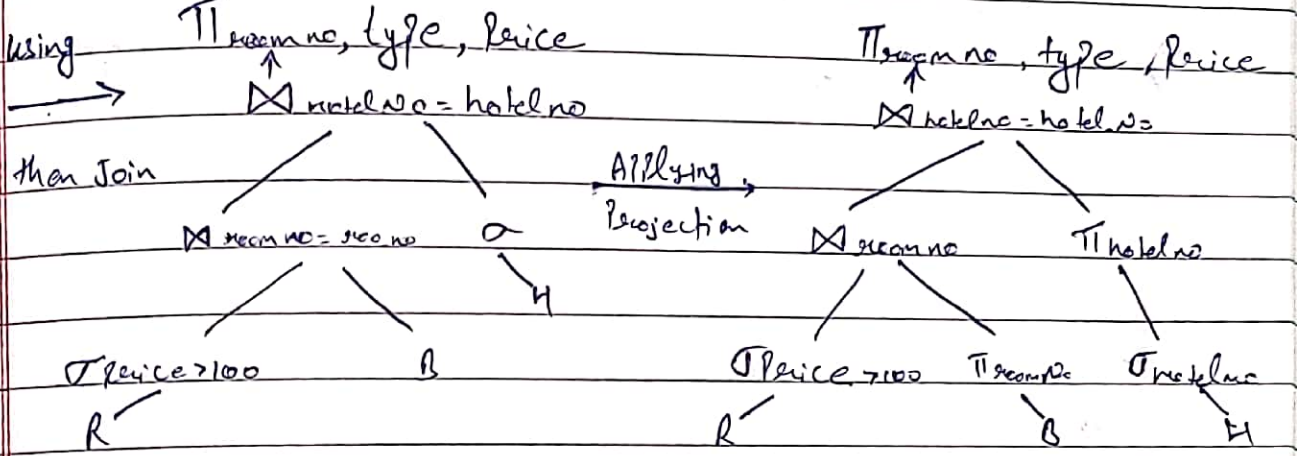
P3: Π_{Price} (Room)
Using Sorting
As price is not Pk [nBlock (R) + nBlockR * $\log_2$ (nBlockR)
= 50 + 50 x $\log_2$ 50 = 350
Using hashing
SC Price(Room) = 10000/50 = 200
no = 1 (we need 1 block for SC Price)
nBlocks (Room) + nB = 51

P5: $\Pi_{\text{hotelNo, Price}}$ (Room)
Using Sorting = 350

using hashing : Hotel type has 51, price no. 51
So overall 51

④ =





⑤ =

If only left side of join is allowed to be something previous join, it forms left deep tree. Similarly if right side of join is allowed to form previous join, it forms right deep tree.
Both these trees are called linear Algebra trees
Trees neither of these are Non linear Algebra trees