1. Low because it will only have either ‘M’ or ‘F’ (I do not know if the system has any other options but for now I am guessing it is this two. Even if there are other options, there will only be one or two more so the data sparsity for that column is still low).

2. I will create Hash indexes because they are good for equality conditions. Hash indexes will be created for EMP\_AREACODE since it is in the Equality condition and it has high Data Sparsity.

CREATE INDEX emp\_arcode\_idx

ON EMPLOYEE (EMP\_AREACODE);

3.

4. The Data Sparsity for EMP\_DOB will be low as it only accounts for the year which will only have a range of 50-70 at most.

6. It will do an Index scan at conditional statement like this which also has low data sparsity is searched directly by the DBMS using the index in the column.

7. Since there is no table statistics I think the DBMS will use the Automatic Query Optimization mode.

8. If the P\_PRICE column is indexed the DBMS will most likely use INDEX access. Otherwise it will go for a Table scan.

9. P\_PRICE has high data sparsity since price can be anything (any numeric value).

11. LINE\_UNITS has high sparsity as again, it can be any number of units,

15. P\_QOH (quantity) and P\_PRICE have high data sparsity

17. I would use Hash Indexes as the data sparsity of V\_STATE is not low since the range of values is 50 but Hash indexing is good for Equality Conditions.

CREATE INDEX v\_name\_idx

ON VENDOR (V\_NAME);

18. It could be anything in a low range because it is looking for only one state “TN” specifically which does not even show up that often in the table.

19. DBMS will do a Table access as the V\_STATE column will not indexed because it has low data sparsity but using row IDs the operation could be performed.

20.

21. I would use the Index Optimizer hint.

SELECT /\* + INDEX (P\_QOH\_NDX) \*/\*

FROM PRODUCT

WHERE P\_QOH < P\_MIN;

22. I would recommend Indexes on V\_CODE, V\_AREACODE, and P\_PRICE.

25. I would recommend index on P\_CODE.

CREATE INDEX p\_code\_idx

ON PRODUCT (P\_CODE);

27. since equality conditions should be written first,

|  |  |
| --- | --- |
| SELECT | P\_CODE, P\_DESCRIPT, P\_QOH, P\_PRICE, V\_CODE |
| FROM | PRODUCT |
| WHERE | P\_MIN = P\_REORDER AND P\_REORDER = 50 AND P\_QOH < P\_MIN |
| ORDER BY | P\_QOH; |

29. First, MAX(LINE\_UNITS\*LINE\_PRICE) column a name and then, index the CUS\_CODE column. Also, use INNER JOIN for both of the joins to be more specific.

31. Indexes on CUS\_CODE would be recommended as one is enough for the CUSTOMER table because it is also the Primary Key.

32.

CREATE INDEX cus\_code\_idx

ON CUSTOMER (CUS\_CODE);

SELECT CUS\_CODE, MAX (LINE\_UNITS \* LINE\_PRICE)

FROM CUSTOMER INNER JOIN INVOICE INNER JOIN LINE

WHERE CUS\_AREACODE = ‘615’

GROUP BY CUS\_CODE;