Prompting Guidelines
Index

1. General Guidelines

a. Use clear and concise natural language queries.
b. Specify relevant attributes when necessary.
c. If requesting filtered data, mention the conditions explicitly.
d. Use date-specific queries carefully, ensuring a valid format (**YYYY-MM-DD**).
e. Include required aggregations (**sum**, **count**, **average**) where applicable.
f. Avoid vague queries like "Get me the data"; be specific about what you need.
g. Structure queries logically to improve readability and execution.
h. Limit data retrieval by using appropriate **WHERE** clauses.
i. Ensure queries adhere to schema constraints and relationships.
j. Specify sorting orders when necessary, using **ORDER BY**.
k. Use **GROUP BY** for aggregated results when dealing with summarized data.
l. Avoid unnecessary subqueries that may impact performance
m. Ensure queries return meaningful insights relevant to the business requirements.

2. Query Types & Examples

2.a. Retrieving Basic Data

Retrieving basic data involves fetching records from tables without complex conditions. These queries help users explore available data and understand the structure of data.

Example:

1. "Show me all SKUs."
2. "List all brands and their categories."
3. "Get all users."
4. "Retrieve all HQ names."
5. "Show all stockists and their IDs."
6. "List all available divisions."
7. "Fetch all sales transactions."
8. "Retrieve all brand names with their IDs."
9. "Get all role names from the Role table."
10. "Show all stockist names."

2.b. Filtering Data

Filtering data allows users to narrow down query results based on specific conditions. Applying filters helps in retrieving only relevant records, reducing unnecessary data processing.

Example:

1. "Get all sales transactions where primary sales is greater than 5000."
2. "Show all stockists in HQ name Kolkata."
3. "Retrieve all users who have login enabled."
4. "Fetch all transactions where status is 'Completed'."
5. "Get all targets where target value is between 1000 and 5000."
6. "List all stockists associated with Division 'D001'."
7. "Show all SKUs that belong to Brand ID 10."
8. "Get all employees who belong to Role ID 3."
9. "Retrieve all sales records for SKU 'ABC123' in HQ 'HQ001'."

2.c. Aggregation Queries

Aggregation queries are used to calculate summary statistics like SUM, COUNT, AVG, MAX, and MIN. These queries provide valuable insights by grouping data based on specific criteria.

Example:

1. "What is the total sales for January 2024?"
2. "Get the average target value per division."
3. "Find the highest sales transaction amount."
4. "Calculate the total number of stockists per HQ."
5. "Show the sum of primary units sold per SKU."
6. "Get the average sales value per stockist."
7. "Find the total sales grouped by brand."
8. "Retrieve the minimum target value recorded."
9. "Get the total sales volume per billing type."

2.d. Time-based Queries

Time-based queries help retrieve data within a specified date or time range. These queries are useful for trend analysis, performance tracking, and historical comparisons.

Example:

1. "Get all sales records for the last 6 months."
2. "Show me target values between 2023-01-01 and 2023-12-31."
3. "Retrieve all transactions from the previous quarter."
4. "Fetch sales performance data for the current fiscal year."
5. "Show monthly revenue trends for the last 12 months."
6. "List the weekly sales breakdown for SKU 'XYZ'."
7. "Get all target records updated in the last 7 days."
8. "Find sales transactions for weekends only."
9. "Fetch quarterly target achievements."
10. "Get all sales records grouped by year."

2.e. User and Role Queries

User and role queries focus on retrieving user-related data, including roles, permissions, and assigned responsibilities. These queries help in access management and organizational reporting.

Example:

1. "List all users with their roles."
2. "Show all managers and their assigned users."
3. "Find users who belong to Role ID 5."
4. "Retrieve login-enabled users with their HQs."
5. "Fetch all employees assigned to multiple roles."
6. "List users along with their reporting managers."
7. "Show user details including their assigned division."
8. "Get the total count of users per role."
9. "Find inactive users who haven't logged in for 6 months."
10. "Retrieve the role names of all employees in HQ 'HQ001'."

2.f. Performance Queries

Performance queries are designed to analyse business efficiency by comparing sales, targets, and other key metrics. They help identify trends, outliers, and areas for improvement.

Example:

1. "What is the sales achievement percentage for each HQ?"
2. "Compare primary sales vs target values for SKU 'ABC123'."
3. "Find the top 5 performing stockists by sales volume."
4. "Show the percentage increase in sales compared to last year."
5. "Retrieve sales data where target achievement is below 80%."
6. "List HQs ranked by highest revenue."
7. "Fetch the stockist with the highest return percentage."
8. "Compare division-wise sales growth over the past 3 years."
9. "Get SKU-wise contribution to total revenue."
10. "Find underperforming regions based on target achievements."

2.g. Historical Data Queries

Historical data queries allow users to retrieve past records for analysis and reporting. They are essential for long-term trend analysis and decision-making based on historical performance.

Example:

1. "Fetch sales data from the last 3 years."
2. "Find historical stock levels of SKU 'XYZ'."
3. "Show revenue trends over the last decade."
4. "List annual sales data for each brand."
5. "Get the number of transactions per month since 2020."
6. "Retrieve stockist sales records for the last 5 years."
7. "Fetch historical employee login records."
8. "Compare sales performance from 2018 to 2023."
9. "Show division-wise revenue breakdown since inception."

2.h. Joining Tables for Related Information

Example:

1. "Show me all stockists and their HQ names."
2. "List SKUs along with their corresponding brand names."

3. Special Considerations

Certain factors, such as case sensitivity, null values, and data accuracy, must be considered when crafting queries. Ensuring these aspects are handled correctly prevents errors and enhances reliability.

Example:

a. "Ensure case sensitivity when filtering by text values."
b. "Validate input dates to avoid incorrect results."
c. "Use indexes to optimize large table queries."
d. "Limit results for better performance in large datasets."
e. "Use joins appropriately when querying multiple tables."
f. "Avoid fetching unnecessary columns to reduce load time."
g. "Check for NULL values in key fields before filtering."
h. "Convert string-based date fields into proper date formats."
i. "Use **DISTINCT** where necessary to avoid duplicate records."
j. "Ensure that primary keys are referenced correctly in queries."

4. Common Mistakes to Avoid

Avoiding common mistakes such as missing filters, incorrect joins, and vague queries can improve query efficiency. Understanding and correcting these errors leads to more precise results.

a. Using generic terms like "get data" without specifying details.
b. Not providing a date range when querying large datasets.
c. Using ambiguous column names that exist in multiple tables.
d. Forgetting to include required filters, such as HQ ID or SKU Code.
e. Requesting too much data at once, leading to performance issues.
f. Using incorrect date formats.
g. Ignoring necessary joins when querying multiple tables.
h. Asking for "all records" instead of setting limits.
i. Confusing similar column names, like id in multiple tables.
j. Not specifying grouping when using aggregate functions.

5. Best Practices

a. Use clear and structured queries with proper keywords.
b. Always specify the table or relevant columns in queries.
c. Use proper filtering conditions to get precise results.
d. Set date ranges explicitly when querying time-based data.
e. Use aggregate functions appropriately with groupings.
f. Keep queries concise and focused on the required data.
g. Ensure accurate joins when combining multiple tables.
h. Limit large queries to avoid performance slowdowns.
i. Use clear column aliases when needed for readability.
j. Validate input values before executing queries.

6. Key Terms:

a. **SELECT**: Used to retrieve specific columns from a table.
b. **WHERE**: Used to filter records based on a condition.
c. **ORDER BY**: Used to sort results in ascending or descending order.
d. **GROUP BY**: Used with aggregate functions to group data.
e. **JOIN**: Used to combine data from multiple tables.
f. **LIMIT**: Restricts the number of rows returned.
g. **DISTINCT**: Removes duplicate records.
h. **HAVING**: Filters groups after aggregation.

7. Best Phrases for Queries:

a. "List all SKUs along with their brand names."
b. "Show total sales for each HQ in the last quarter."
c. "Retrieve all users who have login enabled."
d. "Get primary sales and target values for SKU 'XYZ' in HQ 'HQ001'."
e. "Find top-performing stockists by revenue."
f. "Fetch sales data for the last 12 months grouped by month."
g. "Show me the percentage of sales returns per region."
h. "Retrieve all employee roles along with their assigned divisions."
i. "Find the highest and lowest sales transaction values."
j. "Compare sales achievement percentages across all divisions."

8. Mistakes

a. ❌ "Give me all data" → ✅ "List all sales transactions for January 2024"
b. ❌ "Show users" → ✅ "Show all users with their roles and HQ IDs"
c. ❌ "Get sales" → ✅ "Fetch primary_sales and target_value for SKU 'XYZ' in HQ 'HQ001'"
d. ❌ "List stockists" → ✅ "Show stockist names along with their HQ assignments"
e. ❌ "Retrieve targets" → ✅ "Get target_value and target_units for Division 'D001' in Q1 2024"
f. ❌ "Sales of SKU ABC" → ✅ "Fetch primary_sales and primary_units for SKU 'ABC' in the last quarter"
g. ❌ "Find user queries" → ✅ "Show all user queries from ConversationLog in the last 30 days"