Important Note:

JOB_ID: always add node.job_id = '2ea13652-fd4c-4cc1-9e6f-fdda67a844d5' in WHERE clause for every query generated, job_id and value should not be missed in query, Ensure job_id is not placed after the RETURN statement or any subsequent part of the query.

BACKTIC:always add ` ` Backtic to Node Label and Edge Label, and node property of `$$Name$$`, don't add backtick to job_id.

DATE: When you use of datetime, Always use datetime('2023-01-01T00:00:00'), but don't use datetime(w.`date property`), For any queries involving a "between" condition, ensure the WHERE clause includes both a starting and ending time. The datetime comparison should use this format: AND wo.`date property` >= datetime('YYYY-MM-DDTHH:MM:SS') AND wo.`date property` <= datetime('YYYY-MM-DDTHH:MM:SS'), For questions involving "after" or "before," the comparison should include all dates from the mentioned date onward or backward, respectively. date property are present inside Node properties. always consider date, not year or month.

Single MATCH Clause: Always combine multiple relationships in a single MATCH clause. The relationships should flow naturally from one node to another, without breaking them into separate MATCH clauses.Always ensure that the relationships in a query are part of the same MATCH chain, wherever logically possible. Avoid creating new MATCH clauses for related nodes unless absolutely necessary.

For example:

Correct:

```
MATCH (nodeLabel1:`Label1`)-[relationshipType1:`TYPE1`]-(nodeLabel2:`Label2`)-[relationshipType2:`TYPE2`]-(nodeLabel3:`Label3`)-[relationshipType3:`TYPE3`]-(nodeLabel4:`Label4`)

MATCH (nodeLabel2)-[relationshipType4:`TYPE4`]-(nodeLabel5:`Label5`)-[relationshipType5:`TYPE5`]-(nodeLabel6:`Label6`)
```

Incorrect:

```
MATCH (nodeLabel1:`Label1`)-[relationshipType1:`TYPE1`]-(nodeLabel2:`Label2`)-[relationshipType2:`TYPE2`]-(nodeLabel3:`Label3`)-[relationshipType3:`TYPE3`]-(nodeLabel4:`Label4`)

MATCH (nodeLabel2:`Label2`)-[relationshipType4:`TYPE4`]-(nodeLabel5:`Label5`)-[relationshipType5:`TYPE5`]-(nodeLabel6:`Label6`)
```

VARIABLE: When generating Cypher queries, ensure that each variable has a unique name based on its role (either as a node or a relationship). Avoid reusing the same variable name for both nodes and relationships, as this will cause a type mismatch error. For example: If you define a node as (o:`Order`), do not reuse o` as a relationship variable in the same query. Always use distinct and descriptive variable names for nodes and relationships. always create veriable for nodes and egdes

RETURN: When generating Cypher queries, always create descriptive aliases for returned properties. Use the following format for aliasing: Return properties with the pattern: nodeLabel.`Node Property` AS `Alias Name` ,For example, if returning the `$$Name$$` property of a `Product` node, alias it as Product Name. Make sure to follow this pattern consistently for all returned values. Also avoid recalculating expressions in the RETURN clause. Use the aliases defined in the WITH clause instead of repeating the logic in the RETURN clause.

WHERE Clause Placement: Always place the WHERE clause directly after the first MATCH statement. Include all conditions and job_id within the same WHERE clause. For example, ensure that conditions for job_id are not repeated or split across different WHERE clauses, Do not generate multiple WHERE clauses for the same query. Combine all necessary conditions into one unified WHERE clause following the first MATCH clause. also When Comparing with Node Name always use aliase.`$$Name$$` and when comparing with Node property always use aliase.`Node Property`, do not use both at once like aliase.`Node Property`.`$$Name$$`. Do not add match relatonship in where clause.

TOLOWER:For any OpenCypher query involving string comparisons, always apply toLower(node.`Node Property Name`) to both the node property and the comparison string. This ensures consistent case-insensitive behavior. For all string data, such as `$$Name$$`, `status` or other text properties, use the following formats:

Here are examples:

MATCH (c:`Customer`)

WHERE toLower(c.`$$Name$$`) CONTAINS 'searchString'

Make sure to include toLower(node.`Node Property Name`) every time you use CONTAINS or = with string data to enforce case-insensitive matching. This formatting is required for every string property in the query.

ROUND: Do not apply rounding to any values in the query. Simply return the values as they are, and do not specify decimal places in the query.

PERCENT: When calculating percentage, conditions regarding job_ship_qty should be used in "With" clause. Do not add job_ship_qty compare in where clause.Always use COUNT(w) for the total count, and COUNT(CASE WHEN ...) for conditional counts, and perform percentage calculations with toFloat() inside the same WITH clause.

Example:

```
MATCH (wo:`Work Order`)
WHERE wo.job_id = '2ea13652-fd4c-4cc1-9e6f-fdda67a844d5'
  AND wo.`ship_date` >= datetime('2024-01-01T00:00:00')
  AND wo.`ship_date` <= datetime('2024-12-31T23:59:59')
WITH
  COUNT(CASE WHEN wo.`job_ship_qty` > 4 THEN 1 END) AS jobs_gt_4,
  COUNT(wo) AS total_jobs
RETURN
  toFloat(jobs_gt_4) / toFloat(total_jobs) * 100 AS percentage_jobs_gt_4;
```

WITH Clause: Always use these types of conditions such as node.node_property = value or node.node_property < value in the WITH clause for counting or conditional expressions, Do not include these types of conditions in the WHERE clause. Correct Example: WITH COUNT(CASE WHEN wo.`job_ship_qty` < 6 THEN 1 END) or COUNT(CASE WHEN wo.`job_ship_qty` = 1 THEN 1 END) Incorrect Example: WHERE wo.`job_ship_qty` = 1 or AND wo.`job_ship_qty` < 6

USE_IT: Ensure that any text inside double quotes " ", such as customer names or other identifiers, is used exactly as it appears, including capitalization and spacing, when generating the query. The generated query should not modify or alter the content inside the double quotes in any way.

GROUP BY: When generating Cypher queries, do not use the GROUP BY clause. Instead, achieve grouping by using the WITH clause to specify the grouping keys and any aggregation functions. The WITH clause should be used to pass the grouped data to subsequent parts of the query.

EDGES AND NODE: when there is a question related to total number of Nodes or Edges, alwasy create a query to count the number of Nodes or Edges from data set using job_id Example: MATCH (n) WHERE n.job_id = ' ' RETURN count(n) AS total_nodes; or MATCH ()-[r]-() WHERE r.job_id = ' ' RETURN COUNT(r) AS total_edges;

PERCENTAGE: When calculating the percentage, avoid using any conditional checks inside the `WHERE` clause.

Example: WHERE toLower(node.`node_property`) CONTAINS 'node_property_value'.Part (node): TIRE CAGE: always take 'Part Description' as lower case in WHERE Clause, Example: WHERE toLower(p.'part_description') CONTAINS 'tire cage', and always use this relationship "Node `Work Order` is connected to Node

`Part` via Edge `produces`," for tire cage.CALIBRATION JOBS: Calibration jobs ensure measuring equipment is accurate and meets required standards Always take calibration from Node `Part`, `part_description` and always take part_description in toLower(), create a relation as 'Node `Customer` is connected to Node `Sales Order` via Edge `places`,Node `Sales Order` is connected to Node `Work Order` via Edge `results to`, Node `Work Order` is connected to Node `Part` via Edge `produces`',do not use COUNT(CASE) for calibration.,Tire Cage: always take 'part_description' as in Lower case as toLower(`part_description`) and create a relation as "Node `Work Order` is connected to Node `Part` via Edge `produces`, "When creating a relationship with Part use this relationship "Node `Work Order` is connected to Node `Part` via Edge `produces`,"Revenue Percent: When calculating percentage of revenue for part number, use relationship "Node `Work Order` is connected to Node `Part` via Edge `produces`" and Ensure that the query calculates the minimum and maximum unit price over the given time period. and The result should return the percentage increase between the minimum and maximum unit prices, formatted as a percentage., Sales Partner (node): SALES PARTNER:sales partner should have a relationship like "Node `Sales Partner` is connected to Node `Quote` via Edge `creates`,Node `Quote` is connected to Node `Sales Order` via Edge `converted to`, Node `Sales Order` is connected to Node `Work Order` via Edge `results to`,", Work Order (node): REVENUE: Revenue is the total money a company earns from sales before subtracting costs. Always take revenue from node `Work Order`, If required then create a relationship with node `Work Order`,.VALUE STREAM: value stream should always taken as Node property of `Work Order` Node, not from the `Value Stream` Node, value stream is a property of Work Order not a Node, for value stream always create a match as MATCH (wo.`Work Order`). Do not create a relationship with other node for value streamLICENSE PROGRAM: program should always taken as Node property of `Work Order` Node. for program always create a match as MATCH (wo.`Work Order`). Do not create a relationship with other node for license program., results to (edge): RESULT TO: Always take this edge 'result to' inside backtick, Example: `result to`, Order Date ('property'): 'Part Description' should be taken as lower, Example: toLower(p.'part_description'), PO ('property'): Whenever Sales value or Revenue is asked for in the question make use of this property., Terms ('property'): VALUE STREAM: value stream property of node Work Order should be taken for Value Stream related question, only Work Order Node should be taken for this, Do not create relationship with other nodeExample: MATCH (w.`Work Order`) only this node should be there, Terms Description ('property'): 'Market Description' should be taken in lower case, example toLower(w.market_description) in WHERE Clause, Order Line ('property'): 'Part Description' should be taken as lower, Example: toLower(p.'part_description'), Revenue ('property'): Make use of this property from Work Order node to answer question on Revenue (ex: Total sales, Total Revenue, etc) is asked for in the question make use of this property., Value Stream ('property'): VALUE STREAM: value stream property of node

Work Order should be taken for Value Stream related question, only Work Order Node should be taken for this, Do not create relationship with other nodeExample: MATCH (w.`Work Order`) only this node should be there, Market Description ('property'): 'Market Description' should be taken in lower case, example toLower(w.market_description) in WHERE ClauseWhen calculating revenue contribution for any specific market__description usie CONTAINS filter on the market__description field. and SUM(CASE WHEN ...) for conditional sum only inside WITH clause., Ship Date ('property'): USE DATE: Alwasy consider this date while generating the query, Do not use year or month, Job Ship Qty ('property'): JOB SHIP QTY: Always use this type conditions job_ship_qty inside WITH clause, do not use it inside WHERE clause. Example: WITH COUNT(CASE WHEN wo.`job_ship_qty` = 1 THEN 1 END), CONDITION:Always use 'mil' instead of 'military' while using CONTAINS.LATE SHIPMENTS:lateshipment should always taken as Node properties of `Work Order` Node, Always use 'Late' or 'On Time' when querying for Late shipments.RETURN:When writing Cypher queries, avoid recalculating expressions in the RETURN clause. Use the aliases defined in the WITH clause instead of repeating the logic in the RETURN clause. This ensures the query remains efficient and avoids redundant computations.WITH:when querying using WITH do not include any aliase with that EXAMPLE: WITH wo,.Date from: Always consider w.`ship_date` of node `Work Order` when dealing with Date range.If user asks sales data consider sales persons sales in answer.