

Few_Shots.py

example_list = [

{

"input": "How many bookings are there for zone code",

"output": ""Thought: I need the zone code to write the SQL query

Action: ask_for_parameter

Action Input: "zoneCode"

Observation: I can now use this to parameter to the Question.

Thought: I can now query the database

Action: sql_db_query

Action Input: SELECT

count(BKG_NO) AS Total_Bookings

FROM

BKG_JO_INTERFACE BKG,

CAP_GEO_SCOPE CAP1 ,

CAP_GEO_SCOPE CAP2

WHERE

CAP1.GEO_SCOPE_SEQ = BKG.POL_TERMINAL_SEQ

AND CAP2.GEO_SCOPE_SEQ = BKG.SHIPPER_CITY_SEQ

AND ISNULL(BKG.JO_STATUS,'N') = 'N'

AND CAP2.ZONE_CODE = 'USNC'

Observation: [('USNC',2380)]

Thought: I now have the required data.

Final Answer: There are total 2380 bookings in the zone USNC. ""

},

{

"input": "Which are the top 5 busiest routes",

"output": ""Thought: I need the onDate to write the SQL query

Action: ask_for_parameter

Action Input: "onDate"

Observation: I can now use this to parameter to the Question.

Thought: I can now query the database

Action: sql_db_query

Action Input: SELECT

frm_scope.GEO_SCOPE_CODE AS FROM_LOCATION,

to_scope.GEO_SCOPE_CODE AS TO_LOCATION,

CAST(jh.JOB_ORDER_DATE AS DATE) AS JOB_DATE,

COUNT(*) AS job_count

FROM IML_JOB_ORDER_HEADER jh

JOIN CAP_GEO_SCOPE frm_scope

ON jh.JOB_ORDER_FRMLOC = frm_scope.GEO_SCOPE_SEQ

JOIN CAP_GEO_SCOPE to_scope

ON jh.JOB_ORDER_TOLOC = to_scope.GEO_SCOPE_SEQ

WHERE CAST(jh.JOB_ORDER_DATE AS DATE) = '2023-04-11'

GROUP BY

frm_scope.GEO_SCOPE_CODE,

to_scope.GEO_SCOPE_CODE,

CAST(jh.JOB_ORDER_DATE AS DATE)

ORDER BY job_count DESC

OFFSET 0 ROWS FETCH NEXT 5 ROWS ONLY;

Observation: ...

Final Answer: final result of the query""

},

{

"input": "Which are the top 5 busiest routes in April 2023?",

"output": ""Thought: I need the date range to write the SQL query

Action: ask_for_parameter

Action Input: "To Date" and "From Date"

Observation: I can now use this to parameter to the Question.

Thought: I can now query the database

Action: sql_db_query

Action Input: SELECT

frm_scope.GEO_SCOPE_CODE AS FROM_LOCATION,

to_scope.GEO_SCOPE_CODE AS TO_LOCATION,

FORMAT(jh.JOB_ORDER_DATE, 'yyyy-MM') AS JOB_MONTH,

COUNT(*) AS job_count

FROM IML_JOB_ORDER_HEADER jh

JOIN CAP_GEO_SCOPE frm_scope

ON jh.JOB_ORDER_FRMLOC = frm_scope.GEO_SCOPE_SEQ

JOIN CAP_GEO_SCOPE to_scope

ON jh.JOB_ORDER_TOLOC = to_scope.GEO_SCOPE_SEQ

WHERE jh.JOB_ORDER_DATE >= '2023-04-01'

AND jh.JOB_ORDER_DATE < '2023-05-01'

GROUP BY

frm_scope.GEO_SCOPE_CODE,

```
to_scope.GEO_SCOPE_CODE,  
  
FORMAT(jh.JOB_ORDER_DATE, 'yyyy-MM')  
  
ORDER BY job_count DESC  
  
OFFSET 0 ROWS FETCH NEXT 5 ROWS ONLY;
```

Observation: ...

Final Answer: final result of the query""

},

{

"input": "Select Top 10 Record from Export side (Booking) Empty Booking that has Pending Job order for the zone",

"output": ""Thought: I need the zone code to write the SQL query

Action: ask_for_parameter

Action Input: "USNC"

Observation: I can now use this zone code to parameterize the query.

Thought: I can now query the database.

Action: sql_db_query

Action Input: SELECT TOP (10)

```
BL_ID,  
  
EQUIPMENT_NO,  
  
VESSEL_CODE,  
  
VOYAGE_SML,  
  
CARGO_TYPE,  
  
ARRIVAL_DATE,  
  
ARRIVAL_TIME,  
  
DATEADD(MINUTE, NEW_ARRIVAL_DATE, ARRIVAL_DATE) AS NEW_DATE,  
  
EQ_SIZE,
```

```

EQ_TYPE,

TRANS_MODE,

CUST_HOLD,

ZONE_CODE,

POD_CODE,

POD_TYPE,

CASE

    WHEN POD_TYPE = 'PT' THEN (

        SELECT port_name FROM cap_port WHERE port_code = POD_CODE

    )

    WHEN POD_TYPE IN ('DP', 'TM', 'RR') THEN (

        SELECT terminal_depot_name FROM cap_terminal_depot WHERE
terminal_depot_code = POD_CODE

    )

    WHEN POD_TYPE = 'CT' THEN (

        SELECT city_name FROM cap_city WHERE city_code = POD_CODE

    )

END AS POD_NAME,

DESTINATION_CODE,

DESTINATION_TYPE,

CASE

    WHEN DESTINATION_TYPE = 'PT' THEN (

        SELECT port_name FROM cap_port WHERE port_code = DESTINATION_CODE

    )

    WHEN DESTINATION_TYPE IN ('DP', 'TM', 'RR') THEN (

```

```

        SELECT terminal_depot_name FROM cap_terminal_depot WHERE
terminal_depot_code = DESTINATION_CODE

    )

    WHEN DESTINATION_TYPE = 'CT' THEN (

        SELECT city_name FROM cap_city WHERE city_code = DESTINATION_CODE

    )

END AS DESTINATION_NAME,

CASE

    WHEN POD_STATE_CODE IS NOT NULL THEN ',' + POD_STATE_CODE

    ELSE POD_STATE_CODE

END AS POD_STATE_CODE,

CASE

    WHEN DST_STATE_CODE IS NOT NULL THEN ',' + DST_STATE_CODE

    ELSE DST_STATE_CODE

END AS DST_STATE_CODE,

CONSIGNEE_ADDRESS,

APPOINTMENT_DATE,

APPOINTMENT_TIME,

DATEADD(MINUTE, NEW_APPOINTMENT_DATE, APPOINTMENT_DATE) AS
NEW_APPOINTMENT_DATE,

SCH_DROP_DATE,

SCH_DROP_TIME,

DATEADD(MINUTE, NEW_SCH_DROP_DATE, SCH_DROP_DATE) AS
NEW_SCH_DROP_DATE

FROM (

    SELECT DISTINCT

        BL.BL_ID,

```

BL.EQUIPMENT_NO,
 BL.VESSEL_CODE,
 BL.VOYAGE_SML,
 BL.CARGO_TYPE,
 BL.ARRIVAL_DATE,
 BL.ARRIVAL_TIME,
 FLOOR(BL.ARRIVAL_TIME / 100) * 60 + (BL.ARRIVAL_TIME % 100) AS
 NEW_ARRIVAL_DATE,
 BL.EQ_SIZE,
 BL.EQ_TYPE,
 BL.TRANS_MODE,
 BL.CUST_HOLD,
 CAP2.ZONE_CODE,
 CAP1.GEO_SCOPE_CODE AS POD_CODE,
 CAP1.GEO_SCOPE_TYPE AS POD_TYPE,
 CAP2.GEO_SCOPE_CODE AS DESTINATION_CODE,
 CAP2.GEO_SCOPE_TYPE AS DESTINATION_TYPE,
 CAP1.STATE_CODE AS POD_STATE_CODE,
 CAP2.STATE_CODE AS DST_STATE_CODE,
 BL.CONSIGNEE_ADDRESS,
 BL.APPOINTMENT_DATE,
 BL.APPOINTMENT_TIME,
 FLOOR(BL.APPOINTMENT_TIME / 100) * 60 + (BL.APPOINTMENT_TIME % 100) AS
 NEW_APPOINTMENT_DATE,
 BL.SCH_DROP_DATE,
 BL.SCH_DROP_TIME,

```

        FLOOR(BL.SCH_DROP_TIME / 100) * 60 + (BL.SCH_DROP_TIME % 100) AS
NEW_SCH_DROP_DATE,

        BL.EQUIPMENT_LINE_NO

FROM BL_JO_INTERFACE BL

INNER JOIN CAP_GEO_SCOPE CAP1 ON CAP1.GEO_SCOPE_SEQ = BL.POD_TERMINAL_SEQ

INNER JOIN CAP_GEO_SCOPE CAP2 ON CAP2.GEO_SCOPE_SEQ = BL.CONSIGNEE_CITY_SEQ

WHERE ISNULL(BL.JO_STATUS, 'N') = 'N'

        AND CAP2.ZONE_CODE = 'USNC'

) AS T;

```

Observation: List 10 entries with selected columns.

Final Answer: Here are the top 10 records for export side booking empty bookings with pending job orders for the USNC zone.""""

},

{

"input": "Give top 5 busiest route in shipping domain based on job order as of today",

"output": ""Thought: I need the date to write the SQL query.

Action: ask_for_parameter

Action Input: "today's date"

Observation: I can now use today's date to parameterize the query.

Thought: I can now query the database.

Action: sql_db_query

Action Input: SELECT

```

        frm_scope.GEO_SCOPE_CODE AS FROM_LOCATION,

        to_scope.GEO_SCOPE_CODE AS TO_LOCATION,

        CAST(jh.JOB_ORDER_DATE AS DATE) AS JOB_DATE,

        COUNT(*) AS job_count

```



```

FROM IML_JOB_ORDER_HEADER jh

JOIN CAP_GEO_SCOPE frm_scope

    ON jh.JOB_ORDER_FRMLOC = frm_scope.GEO_SCOPE_SEQ

JOIN CAP_GEO_SCOPE to_scope

    ON jh.JOB_ORDER_TOLOC = to_scope.GEO_SCOPE_SEQ

WHERE CAST(jh.JOB_ORDER_DATE AS DATE) = '2023-04-11'

GROUP BY frm_scope.GEO_SCOPE_CODE, to_scope.GEO_SCOPE_CODE,
CAST(jh.JOB_ORDER_DATE AS DATE)

ORDER BY job_count DESC

OFFSET 0 ROWS FETCH NEXT 5 ROWS ONLY;

```

Observation: List the 5 busiest routes based on job orders.

Final Answer: As of today, the top 5 busiest shipping routes based on job orders are:

1. SIN → SHA with 120 job orders
2. LAX → HKG with 115 job orders
3. HAM → DXB with 102 job orders
4. NYC → LON with 95 job orders
5. SGP → TYO with 89 job orders."""

},

{

"input": "Select Top 10 record of Import (BL) side that are In Transit with Truck or Rail",

"output": ""Thought: I need the zone code to write the SQL query

Action: ask_for_parameter

Action Input: "zoneCode"

Observation: I can now use this to parameter to the Question.

Thought: I can now query the database

Action: sql_db_query

Action Input:

```
SELECT TOP (10)

BL_ID,

EQUIPMENT_NO,

CARGO_TYPE,

JOB_ORDER_NUM,

VESSEL_CODE,

VOYAGE_SML,

GEO_SCOPE_CODE,

GEO_SCOPE_TYPE,

CASE

    WHEN GEO_SCOPE_TYPE = 'PT' THEN

        (SELECT port_name FROM cap_port WHERE port_code = GEO_SCOPE_CODE)

    WHEN GEO_SCOPE_TYPE IN ('DP', 'TM', 'RR') THEN

        (SELECT terminal_depot_name FROM cap_terminal_depot WHERE terminal_depot_code =
GEO_SCOPE_CODE)

    WHEN GEO_SCOPE_TYPE = 'CT' THEN

        (SELECT city_name FROM cap_city WHERE city_code = GEO_SCOPE_CODE)

END AS GEO_SCOPE_NAME,

JOB_ORDER_STATUS,

ARRIVAL_DATE,

ARRIVAL_TIME,

INTERMODEL_FLAG,

CAST(DateAdd(Minute, New_Arrival_Date, Arrival_Date) AS Datetime) AS New_Date,

APPOINTMENT_DATE,

APPOINTMENT_TIME,
```

```

CAST(DATEADD(MINUTE, NEW_APPOINTMENT_DATE, APPOINTMENT_DATE) AS DATETIME) AS
NEW_DATE1,

EQ_SIZE,

EQ_TYPE,

PARTNER_NAME,

CONTRACT_NO,

EXPECT_STAT,

ETA,

FROM_LOC,

FROM_GEO_SCOPE_TYPE,

CASE

    WHEN FROM_GEO_SCOPE_TYPE = 'PT' THEN

        (SELECT port_name FROM cap_port WHERE port_code = FROM_LOC)

    WHEN FROM_GEO_SCOPE_TYPE IN ('DP', 'TM', 'RR') THEN

        (SELECT terminal_depot_name FROM cap_terminal_depot WHERE terminal_depot_code =
FROM_LOC)

    WHEN FROM_GEO_SCOPE_TYPE = 'CT' THEN

        (SELECT city_name FROM cap_city WHERE city_code = FROM_LOC)

END AS FROM_LOC_NAME,

TO_LOC,

TO_GEO_SCOPE_TYPE,

CASE

    WHEN TO_GEO_SCOPE_TYPE = 'PT' THEN

        (SELECT port_name FROM cap_port WHERE port_code = TO_LOC)

    WHEN TO_GEO_SCOPE_TYPE IN ('DP', 'TM', 'RR') THEN

        (SELECT terminal_depot_name FROM cap_terminal_depot WHERE terminal_depot_code =
TO_LOC)

```

```

    WHEN TO_GEO_SCOPE_TYPE = 'CT' THEN

        (SELECT city_name FROM cap_city WHERE city_code = TO_LOC)

    END AS TO_LOC_NAME,

    RPT_DATETIME,

    ALERT,

    TRANS_MODE,

    CASE

        WHEN FROM_STATE_CODE IS NOT NULL THEN ',' + FROM_STATE_CODE

        ELSE FROM_STATE_CODE

    END AS FROM_STATE_CODE,

    CASE

        WHEN TO_STATE_CODE IS NOT NULL THEN ',' + TO_STATE_CODE

        ELSE TO_STATE_CODE

    END AS TO_STATE_CODE,

    CASE

        WHEN POD_STATE_CODE IS NOT NULL THEN ',' + POD_STATE_CODE

        ELSE POD_STATE_CODE

    END AS POD_STATE_CODE,

    ACT_ARRIVAL_TIME,

    TO_ACTIVITY,

    FRM_ACTIVITY,

    CASE

        WHEN ACT_ARRIVAL_TIME IS NOT NULL THEN

            (SELECT ACTIVITY_LONG_DESCR FROM EQT_ACTIVITY_CODE WHERE ACTIVITY_CODE =

TO_ACTIVITY)

        ELSE

```

```
(SELECT ACTIVITY_LONG_DESCR FROM EQT_ACTIVITY_CODE WHERE ACTIVITY_CODE =  
FRM_ACTIVITY)
```

```
END AS EXP_STATUS,
```

```
ORIGIN_DEST_ADDRESS,
```

```
ZONE_CODE
```

```
FROM (
```

```
SELECT DISTINCT
```

```
BL.BL_ID,
```

```
BL.EQUIPMENT_NO,
```

```
BL.CARGO_TYPE,
```

```
JOB.JOB_ORDER_NUM,
```

```
BL.VESSEL_CODE,
```

```
BL.VOYAGE_SML,
```

```
CAP.GEO_SCOPE_CODE,
```

```
CAP.GEO_SCOPE_TYPE,
```

```
JOB.JOB_ORDER_STATUS,
```

```
BL.ARRIVAL_DATE,
```

```
BL.ARRIVAL_TIME,
```

```
BL.INTERMODEL_FLAG,
```

```
FLOOR(ARRIVAL_TIME/100)*60 + (ARRIVAL_TIME % 100) NEW_ARRIVAL_DATE,
```

```
BL.APPOINTMENT_DATE,
```

```
BL.APPOINTMENT_TIME,
```

```
FLOOR(APPOINTMENT_TIME/100)*60 + (APPOINTMENT_TIME % 100)
```

```
NEW_APPOINTMENT_DATE,
```

```
BL.EQ_SIZE,
```

```
BL.EQ_TYPE,
```

```

BIZ.PARTNER_NAME,

JOB.CONTRACT_NO,

" EXPECT_STAT,

IJOAD.SCH_ARRIVAL_TIME ETA,

CASE WHEN CAP1.GEO_SCOPE_TYPE='PO' THEN CAP1.CITY_CODE ELSE CAP1.GEO_SCOPE_CODE
END AS FROM_LOC,

CASE WHEN CAP1.GEO_SCOPE_TYPE='PO' THEN 'CT' ELSE CAP1.GEO_SCOPE_TYPE END AS
FROM_GEO_SCOPE_TYPE,

CASE WHEN CAP2.GEO_SCOPE_TYPE='PO' THEN CAP2.CITY_CODE ELSE CAP2.GEO_SCOPE_CODE
END AS TO_LOC,

CASE WHEN CAP2.GEO_SCOPE_TYPE='PO' THEN 'CT' ELSE CAP2.GEO_SCOPE_TYPE END AS
TO_GEO_SCOPE_TYPE,

" RPT_DATETIME,

" ALERT,

BL.TRANS_MODE,

CAP1.STATE_CODE AS FROM_STATE_CODE,

CAP2.STATE_CODE AS TO_STATE_CODE,

CAP.STATE_CODE AS POD_STATE_CODE,

IJOAD.ACT_ARRIVAL_TIME AS ACT_ARRIVAL_TIME,

IJOAD.TO_ACTIVITY AS TO_ACTIVITY,

IJOAD.FRM_ACTIVITY AS FRM_ACTIVITY,

JOB.ORIGIN_DEST_ADDRESS,

CAP3.ZONE_CODE AS ZONE_CODE

FROM

BL_JO_INTERFACE BL

INNER JOIN IML_JOB_ORDER_CONTAINER JOCONT ON JOCONT.REF_NUM = BL.BL_ID

```

```

        INNER JOIN IML_JOB_ORDER_HEADER JOB ON JOB.JOB_ORDER_SEQ =
JOCONT.JOB_ORDER_SEQ

        INNER JOIN CAP_GEO_SCOPE CAP ON CAP.GEO_SCOPE_SEQ = BL.POD_TERMINAL_SEQ

        INNER JOIN CAP_GEO_SCOPE CAP1 ON CAP1.GEO_SCOPE_SEQ = JOB.JOB_ORDER_FRMLOC

        INNER JOIN CAP_GEO_SCOPE CAP2 ON CAP2.GEO_SCOPE_SEQ = JOB.JOB_ORDER_TOLOC

        INNER JOIN V_BPM_CURRENT_BIZ_PARTNER_LOV BIZ ON BIZ.BIZ_PARTNER_SEQ =
JOB.BIZ_PARTNER_SEQ

        INNER JOIN CAP_GEO_SCOPE CAP3 ON CAP3.GEO_SCOPE_SEQ = BL.CONSIGNEE_CITY_SEQ

        INNER JOIN IML_JOB_ORDER_ARR_DEP IJOAD ON IJOAD.JOB_ORDER_SEQ =
JOB.JOB_ORDER_SEQ

WHERE

        BL.JO_STATUS = 'P'

        AND JOB.JOB_ORDER_STATUS NOT IN ('C','R')

        AND JOB.JOB_ORDER_NUM IN ('ORIGIN_JO1', 'ORIGIN_JO2', 'RAIL_JO_MAINLEG', 'DEST_JO1',
'DEST_JO2')

        AND CAP3.ZONE_CODE = 'USNC'

    ) AS T

ORDER BY BL_ID;

```

Observation: List 10 entries with selected column

Final Answer: Here are the top 10 import records that are in transit with Truck or Rail for the USNC zone.

"""

},

{

"input": "Give top 5 busiest route in shipping domain based on job order as of today",

"output": ""Thought: I need the date to write the SQL query

Action: ask_for_parameter

Action Input: "onDate"

Observation: I can now use this to parameter to the Question.

Thought: I can now query the database

Action: sql_db_query

Action Input:

```
SELECT
    frm_scope.GEO_SCOPE_CODE AS FROM_LOCATION,
    to_scope.GEO_SCOPE_CODE AS TO_LOCATION,
    CAST(jh.JOB_ORDER_DATE AS DATE) AS JOB_DATE,
    COUNT(*) AS job_count
FROM IML_JOB_ORDER_HEADER jh
JOIN CAP_GEO_SCOPE frm_scope
    ON jh.JOB_ORDER_FRMLOC = frm_scope.GEO_SCOPE_SEQ
JOIN CAP_GEO_SCOPE to_scope
    ON jh.JOB_ORDER_TOLOC = to_scope.GEO_SCOPE_SEQ
WHERE CAST(jh.JOB_ORDER_DATE AS DATE) = '2023-04-11'
GROUP BY frm_scope.GEO_SCOPE_CODE, to_scope.GEO_SCOPE_CODE, CAST(jh.JOB_ORDER_DATE
AS DATE)
ORDER BY job_count DESC
OFFSET 0 ROWS FETCH NEXT 5 ROWS ONLY;
```

Observation: List the top 5 busiest routes as of today based on job orders.

Final Answer: Based on the SQL query, the top 5 busiest routes as of today are:

1. SIN → SHA with 120 job orders
2. LAX → HKG with 115 job orders
3. HAM → DXB with 102 job orders
4. NYC → LON with 95 job orders

5. SGP → TYO with 89 job orders.

""

},

{

'input': "Total No of Bill of Lading (BL) pending for Job Order (JO) for Zone USNC",

'output': ""Thought: I need the zone code to write the SQL query

Action: ask_for_parameter

Action Input: "zoneCode"

Observation: I can now use this to parameter to the Question.

Thought: I can now query the database

Action: sql_db_query

Action Input:

SELECT COUNT(DISTINCT BL_ID) AS TOTAL_BL_COUNT

FROM (

SELECT

BL_ID,

EQUIPMENT_NO,

VESSEL_CODE,

VOYAGE_SML,

CARGO_TYPE,

ARRIVAL_DATE,

ARRIVAL_TIME,

DATEADD(MINUTE, NEW_ARRIVAL_DATE, ARRIVAL_DATE) AS NEW_DATE,

EQ_SIZE,

EQ_TYPE,

```

TRANS_MODE,

CUST_HOLD,

ZONE_CODE,

POD_CODE,

POD_TYPE,

CASE

    WHEN POD_TYPE = 'PT' THEN (

        SELECT port_name FROM cap_port WHERE port_code = POD_CODE

    )

    WHEN POD_TYPE IN ('DP', 'TM', 'RR') THEN (

        SELECT terminal_depot_name FROM cap_terminal_depot WHERE terminal_depot_code =
POD_CODE

    )

    WHEN POD_TYPE = 'CT' THEN (

        SELECT city_name FROM cap_city WHERE city_code = POD_CODE

    )

END AS POD_NAME,

DESTINATION_CODE,

DESTINATION_TYPE,

CASE

    WHEN DESTINATION_TYPE = 'PT' THEN (

        SELECT port_name FROM cap_port WHERE port_code = DESTINATION_CODE

    )

    WHEN DESTINATION_TYPE IN ('DP', 'TM', 'RR') THEN (

        SELECT terminal_depot_name FROM cap_terminal_depot WHERE terminal_depot_code =
DESTINATION_CODE

```

```

)

WHEN DESTINATION_TYPE = 'CT' THEN (

    SELECT city_name FROM cap_city WHERE city_code = DESTINATION_CODE

)

END AS DESTINATION_NAME,

CASE

    WHEN POD_STATE_CODE IS NOT NULL THEN ',' + POD_STATE_CODE

    ELSE POD_STATE_CODE

END AS POD_STATE_CODE,

CASE

    WHEN DST_STATE_CODE IS NOT NULL THEN ',' + DST_STATE_CODE

    ELSE DST_STATE_CODE

END AS DST_STATE_CODE,

CONSIGNEE_ADDRESS,

APPOINTMENT_DATE,

APPOINTMENT_TIME,

DATEADD(MINUTE, NEW_APPOINTMENT_DATE, APPOINTMENT_DATE) AS
NEW_APPOINTMENT_DATE,

SCH_DROP_DATE,

SCH_DROP_TIME,

DATEADD(MINUTE, NEW_SCH_DROP_DATE, SCH_DROP_DATE) AS NEW_SCH_DROP_DATE

FROM (

    SELECT DISTINCT

        BL.BL_ID,

        BL.EQUIPMENT_NO,

        BL.VESSEL_CODE,

```

BL.VOYAGE_SML,
BL.CARGO_TYPE,
BL.ARRIVAL_DATE,
BL.ARRIVAL_TIME,
FLOOR(BL.ARRIVAL_TIME / 100) * 60 + (BL.ARRIVAL_TIME % 100) AS NEW_ARRIVAL_DATE,
BL.EQ_SIZE,
BL.EQ_TYPE,
BL.TRANS_MODE,
BL.CUST_HOLD,
CAP2.ZONE_CODE,
CAP1.GEO_SCOPE_CODE AS POD_CODE,
CAP1.GEO_SCOPE_TYPE AS POD_TYPE,
CAP2.GEO_SCOPE_CODE AS DESTINATION_CODE,
CAP2.GEO_SCOPE_TYPE AS DESTINATION_TYPE,
CAP1.STATE_CODE AS POD_STATE_CODE,
CAP2.STATE_CODE AS DST_STATE_CODE,
BL.CONSIGNEE_ADDRESS,
BL.APPOINTMENT_DATE,
BL.APPOINTMENT_TIME,
FLOOR(BL.APPOINTMENT_TIME / 100) * 60 + (BL.APPOINTMENT_TIME % 100) AS
NEW_APPOINTMENT_DATE,
BL.SCH_DROP_DATE,
BL.SCH_DROP_TIME,
FLOOR(BL.SCH_DROP_TIME / 100) * 60 + (BL.SCH_DROP_TIME % 100) AS
NEW_SCH_DROP_DATE,
BL.EQUIPMENT_LINE_NO

```

FROM BL_JO_INTERFACE BL

INNER JOIN CAP_GEO_SCOPE CAP1 ON CAP1.GEO_SCOPE_SEQ = BL.POD_TERMINAL_SEQ

INNER JOIN CAP_GEO_SCOPE CAP2 ON CAP2.GEO_SCOPE_SEQ = BL.CONSIGNEE_CITY_SEQ

WHERE ISNULL(BL.JO_STATUS, 'N') = 'N'

AND CAP2.ZONE_CODE = 'USNC'

) AS T

) AS M;

```

Observation: Total number of Bill of Lading pending for Job Order (JO) for Zone USNC.

Final Answer: The total number of pending Bill of Lading (BL) for Zone USNC is 520.

""""

},

{

'input': "List or Select 10 entries with ZONE CODE, COUNTRY CODE, AREA CODE, DEPOT CODE, and TERMINAL CODE.",

'output': """"Thought: I need the column names for the query

Action: ask_for_parameter

Action Input: "zoneCode"

Observation: I can now use this to parameterize the query.

Thought: I can now query the database

Action: sql_db_query

Action Input:

```

SELECT TOP (10)

[ZONE_CODE],

[COUNTRY_CODE],

[AREA_CODE],

[DEPOT_CODE],

```

[TERMINAL_CODE]

FROM [SML_TEST_2022].[dbo].[CAP_GEO_SCOPE]

Observation: List 10 entries with specified columns.

Final Answer:

| ZONE_CODE | COUNTRY_CODE | AREA_CODE | DEPOT_CODE | TERMINAL_CODE |

|-----|-----|-----|-----|-----|

| USFL | US | FL | RDLNC | |

""

},

{

"input": "list of top 5 vendor with maximum invoice value",

"output": ""Thought: I need the invoice data and vendor details to write the SQL query

Action: ask_for_parameter

Action Input: "None" No specific parameter needed for this query

Observation: I can now query the database

Thought: I can now query the database to get the top 5 vendors based on the maximum invoice value

Action: sql_db_query

Action Input:

SELECT TOP 5

f.PAY_TO_VENDOR,

bpn.PARTNER_NAME,

SUM(f.INVOICE_AMT) AS TOTAL_AMT

FROM

FAP_INVOICE_HDR f

JOIN

BPM_BIZ_PARTNER bp

ON bp.PARTNER_CODE = f.PAY_TO_VENDOR

JOIN

BPM_PARTNER_NAME bpn

ON bp.BIZ_PARTNER_SEQ = bpn.BIZ_PARTNER_SEQ

GROUP BY

f.PAY_TO_VENDOR, bpn.PARTNER_NAME

ORDER BY

TOTAL_AMT DESC;

Observation:

PAY_TO_VENDOR	PARTNER_NAME	TOTAL_AMT
-----	-----	-----
USC001021560	ZNN XPRESS INC.	1,200,000
USC001021561	FAST CARGO INC.	950,000
USC001021562	EXPEDITORS LTD.	850,000
USC001021563	SHIP FAST INC.	800,000
USC001021564	CARGO PARTNERS	750,000

Final Answer: The top 5 vendors with maximum invoice value are as listed. ""

},

{

"input": "Who are the top 5 vendors with the highest number of Job Orders in March 2023?",

"output": ""Thought: I need the job order data and the vendor details for March 2023

Action: ask_for_parameter

Action Input: "Date" Asking for the date range parameter

Observation: I can now use this to query the database

Thought: I can now query the database to get the top 5 vendors based on Job Order count in March 2023

Action: sql_db_query

Action Input:

```
SELECT TOP 5 WITH TIES
FORMAT(jh.JOB_ORDER_DATE, 'yyyy-MM') AS JOB_MONTH,
jh.BIZ_PARTNER_SEQ,
bp.PARTNER_NAME,
COUNT(*) AS JOB_COUNT
FROM IML_JOB_ORDER_HEADER jh
JOIN BPM_PARTNER_NAME bp
ON jh.BIZ_PARTNER_SEQ = bp.BIZ_PARTNER_SEQ
WHERE
bp.RECORD_STATUS = 'A'
AND FORMAT(jh.JOB_ORDER_DATE, 'yyyy-MM') = '2023-03'
GROUP BY
FORMAT(jh.JOB_ORDER_DATE, 'yyyy-MM'),
jh.BIZ_PARTNER_SEQ,
bp.PARTNER_NAME
ORDER BY
JOB_MONTH, JOB_COUNT DESC;
```

Observation:

JOB_MONTH	BIZ_PARTNER_SEQ	PARTNER_NAME	JOB_COUNT
-----	-----	-----	-----
2023-03	986556	ZNN XPRESS INC.	68
2023-03	986557	CARGO MOVERS	60

2023-03	986558	FAST LOGISTICS	50	
2023-03	986559	TRANSIT PRO	45	
2023-03	986560	GLOBAL SHIPPING	40	

Final Answer: The top 5 vendors with the highest number of Job Orders in March 2023 are as listed. ""

},

{

"input": "How many job orders are in process with Job Order Status 's' based on Partner name 'ZNN XPRESS INC.' or Partner Code 'USC000986556'?",

"output": ""Thought: I need the job orders based on the partner details and job order status.

Action: ask_for_parameter

Action Input: "Partner name or Partner Code" Asking for partner details

Observation: I can now query the database

Thought: I can now query the database to count the job orders in process for the specified partner

Action: sql_db_query

Action Input:

SELECT COUNT(*) AS JOB_ORDER_TOTAL_COUNT

FROM (

SELECT DISTINCT

joborderheader.JOB_ORDER_SEQ,

joborderheader.JOB_ORDER_NUM,

bpmbizpartner.PARTNER_CODE,

joborderheader.VERSION,

joborderheader.JOB_ORDER_LEG_TYPE,

joborderheader.JOB_ORDER_TYPE,

joborderheader.JOB_ORDER_STATUS,

joborderheader.JOB_ORDER_COST,

```

    jobborderheader.JOB_ORDER_CURRENCY,
    jobbordercontainer.REF_NUM,
    jobbordercontainer.CONTAINER_NUM,
    jobborderheader.RECORD_STATUS,
    jobborderheader.JOB_ORDER_DATE,
    bpmbizpartner.PARTNER_FUNCTION_TYPE_CODE,
    bpmbizpartner.PARTNER_NAME as PARTNER_NAME,
    jobbordercontainer.EQ_SIZE,
    jobbordercontainer.EQ_TYPE,
    jobbordercontainer.EQUIPMENT_LINE_NO,
    jobbordercontainer.CHASSIS_NUMBER
FROM
    IML_JOB_ORDER_HEADER AS jobborderheader
JOIN
    IML_JOB_ORDER_CONTAINER AS jobbordercontainer
    ON jobborderheader.JOB_ORDER_SEQ = jobbordercontainer.JOB_ORDER_SEQ
JOIN
    V_BPM_CURRENT_BIZ_PARTNER_LOV AS bpmbizpartner
    ON bpmbizpartner.BIZ_PARTNER_SEQ = jobborderheader.BIZ_PARTNER_SEQ
WHERE
    (bpmbizpartner.PARTNER_CODE = 'USC000986556' OR
    bpmbizpartner.PARTNER_NAME like '%ZNN XPRESS INC.%' )
    AND jobborderheader.JOB_ORDER_STATUS = 'S'
) AS subquery;

```

Observation:

```
| JOB_ORDER_TOTAL_COUNT |
```

```
|-----|
```

```
| 183      |
```

Final Answer: There are 183 job orders in process with Job Order Status 's' for the partner 'ZNN XPRESS INC.' or Partner Code 'USC000986556'. ""

```
},
```

```
{
```

```
"input": "Give me (Select) Top 5 repetitive customers based on Bill of Lading (BL)",
```

```
"output": ""Thought: I need the customer details and Bill of Lading (BL) information to write the SQL query
```

Action: ask_for_parameter

Action Input: "None" No specific parameter needed for this query

Observation: I can now query the database

Thought: I can now query the database to get the top 5 repetitive customers based on Bill of Lading (BL)

Action: sql_db_query

Action Input:

```
SELECT TOP 5
```

```
    COUNT(NAME_OF_THE_CUSTOMER) AS CUSTOMER_COUNT,
```

```
    BKGBL_INTERFACE_TYPE,
```

```
    NAME_OF_THE_CUSTOMER
```

```
FROM BKGBL_JO_INTERFACE_ADDRESS
```

```
WHERE BKGBL_INTERFACE_TYPE = 'BL'
```

```
GROUP BY BKGBL_INTERFACE_TYPE, NAME_OF_THE_CUSTOMER
```

```
ORDER BY BKGBL_INTERFACE_TYPE ASC, CUSTOMER_COUNT DESC;
```

Observation:

```
+-----+-----+-----+
```

CUSTOMER_COUNT	BKGBL_INTERFACE_TYPE	NAME_OF_THE_CUSTOMER
7745	BL	GILDAN ACTIVEWEAR
6910	BL	VI VIRGINIA SERVICE CENTER
5140	BL	SANMAR CORPORATION
4945	BL	13 HANESBRAND, INC
4619	BL	STERIGENICS

Final Answer: These are the top 5 repetitive customers based on Bill of Lading (BL):

1. GILDAN ACTIVEWEAR with 7745 entries
2. VI VIRGINIA SERVICE CENTER with 6910 entries
3. SANMAR CORPORATION with 5140 entries
4. 13 HANESBRAND, INC with 4945 entries
5. STERIGENICS with 4619 entries

},

{

"input": "Give me (Select) Top 1 repetitive customer based on Booking (BK)",

"output": ""Thought: I need the customer details and Booking (BK) information to write the SQL query

Action: ask_for_parameter

Action Input: "None" No specific parameter needed for this query

Observation: I can now query the database

Thought: I can now query the database to get the top 1 repetitive customer based on Booking (BK)

Action: sql_db_query

Action Input:

SELECT TOP 1

```

COUNT(NAME_OF_THE_CUSTOMER) AS CUSTOMER_COUNT,

BKGBL_INTERFACE_TYPE,

NAME_OF_THE_CUSTOMER

FROM BKGBL_JO_INTERFACE_ADDRESS

WHERE BKGBL_INTERFACE_TYPE = 'BK'

GROUP BY BKGBL_INTERFACE_TYPE, NAME_OF_THE_CUSTOMER

ORDER BY BKGBL_INTERFACE_TYPE ASC, CUSTOMER_COUNT DESC;

```

Observation:

CUSTOMER_COUNT	BKGBL_INTERFACE_TYPE	NAME_OF_THE_CUSTOMER
21970	BK	GILDAN- CUS. MOVED TO RAMP

Final Answer: The top repetitive customer based on Booking (BK) is GILDAN- CUS. MOVED TO RAMP with 21970 entries."

},

{

"input": "Total No of Booking on 2023-04-11",

"output": ""Thought: I need the booking data for the specified date to write the SQL query

Action: ask_for_parameter

Action Input: "Date" Asking for the specific date parameter

Observation: I can now use this to query the database

Thought: I can now query the database to get the total number of bookings on 2023-04-11

Action: sql_db_query

Action Input:

```

SELECT

'DAY' AS TIME_TYPE,

CAST('2023-04-11' AS DATE) AS TIME_VALUE,

```

```
COUNT(DISTINCT BKG_NO) AS TOTAL_BOOKINGS  
FROM BKG_JO_INTERFACE  
WHERE CAST(JO_DATE AS DATE) = '2023-04-11';
```

Observation:

```
| TIME_TYPE | TIME_VALUE | TOTAL_BOOKINGS |  
|-----|-----|-----|  
| DAY      | 2023-04-11 | 2              |
```

Final Answer: The total number of bookings on 2023-04-11 is 2. ""

},

{

"input": "Total No of Bill of Lading (BL) on Daily Basis",

"output": ""Thought: I need the Bill of Lading data for a specific date to write the SQL query

Action: ask_for_parameter

Action Input: "Date" Asking for the specific date parameter

Observation: I can now use this to query the database

Thought: I can now query the database to get the total number of Bill of Lading on the given date

Action: sql_db_query

Action Input:

```
SELECT  
    'DAY' AS TIME_TYPE,  
    CAST('2023-04-11' AS DATE) AS TIME_VALUE,  
    COUNT(DISTINCT BL_NO) AS TOTAL_BL  
FROM BL_JO_INTERFACE  
WHERE CAST(JO_DATE AS DATE) = '2023-04-11';
```

Observation:

TIME_TYPE	TIME_VALUE	TOTAL_BL	
-----	-----	-----	
DAY	2023-04-11	1	

Final Answer: The total number of Bill of Lading (BL) on 2023-04-11 is 1. """

```
},
{
  "input": "Total No of Invoice received on 2019-04-11",
  "output": ""Thought: I need the invoice data for a specific date to write the SQL query
```

Action: ask_for_parameter

Action Input: "Date" Asking for the specific date parameter

Observation: I can now use this to query the database

Thought: I can now query the database to get the total number of invoices received on the given date

Action: sql_db_query

Action Input:

```
SELECT
  'DAY' AS TIME_TYPE,
  CAST('2019-04-11' AS DATE) AS TIME_VALUE,
  COUNT(*) AS TOTAL_INVOICES
FROM FAP_INVOICE_HDR
WHERE CAST(INVOICE_DATE AS DATE) = '2019-04-11';
```

Observation:

TIME_TYPE	TIME_VALUE	TOTAL_INVOICES	
-----	-----	-----	
DAY	2019-04-11	160	

Final Answer: The total number of invoices received on 2019-04-11 is 160. ""

},

{

"input": "Show Top 2 routes with maximum cost within given date range",

"output": ""Thought: I need the specific date range and route information to write the SQL query

Action: ask_for_parameter

Action Input: "startDate, endDate" Asking for the date range

Observation: I can now use the date range to query the database

Thought: I can now query the database to get the top 2 routes with the maximum cost within the given date range

Action: sql_db_query

Action Input:

SELECT TOP 2

FORMAT(jh.JOB_ORDER_DATE, 'dd-yyyy-MM') AS JOB_DATE,

geo_from.GEO_SCOPE_CODE AS FROM_LOCATION,

geo_to.GEO_SCOPE_CODE AS TO_LOCATION,

COUNT(*) AS JOB_COUNT,

SUM(jh.JOB_ORDER_COST) AS COST

FROM IML_JOB_ORDER_HEADER jh

JOIN CAP_GEO_SCOPE geo_from

ON jh.JOB_ORDER_FRMLOC = geo_from.GEO_SCOPE_SEQ

JOIN CAP_GEO_SCOPE geo_to

ON jh.JOB_ORDER_TOLOC = geo_to.GEO_SCOPE_SEQ

WHERE

CAST(jh.JOB_ORDER_DATE AS DATE) BETWEEN '2023-01-01' AND '2024-01-01'

GROUP BY


```
FORMAT(jh.JOB_ORDER_DATE, 'dd-yyyy-MM'),
```

```
geo_from.GEO_SCOPE_CODE,
```

```
geo_to.GEO_SCOPE_CODE
```

```
ORDER BY
```

```
COST DESC,
```

```
JOB_DATE,
```

```
JOB_COUNT;
```

Observation:

JOB_DATE	FROM_LOCATION	TO_LOCATION	JOB_COUNT	COST
31-2023-03	CXMIA	FCMIZ	1	1111111.0
28-2023-04	USPOM	DTCNC	17	946941.6

Final Answer: The top 2 routes with the maximum cost within the given date range are:

1. CXMIA to FCMIZ with a cost of 1,111,111.0 (on 2023-03-31)
2. USPOM to DTCNC with a cost of 946,941.6 (on 2023-04-28)""

},

{

"input": "Shows top 2 vendors with maximum cost within date range",

"output": ""Thought: I need to find the vendors and their costs for the given date range to write the SQL query

Action: ask_for_parameter

Action Input: "startDate, endDate" Asking for the specific date range

Observation: I can now use this to query the database

Thought: I can now query the database to get the top 2 vendors with maximum cost within the given date range

Action: sql_db_query

Action Input:

```
SELECT TOP 2 WITH TIES
    FORMAT(jh.JOB_ORDER_DATE, 'dd-yyyy-MM') AS JOB_DATE,
    bp.PARTNER_NAME,
    COUNT(*) AS JOB_COUNT,
    SUM(jh.JOB_ORDER_COST) AS COST
FROM IML_JOB_ORDER_HEADER jh
JOIN BPM_PARTNER_NAME bp
    ON jh.BIZ_PARTNER_SEQ = bp.BIZ_PARTNER_SEQ
WHERE
    bp.RECORD_STATUS = 'A'
    AND jh.JOB_ORDER_DATE >= '2023-01-01'
    AND jh.JOB_ORDER_DATE < '2024-01-01'
GROUP BY
    FORMAT(jh.JOB_ORDER_DATE, 'dd-yyyy-MM'),
    jh.BIZ_PARTNER_SEQ,
    bp.PARTNER_NAME
ORDER BY
    COST DESC,
    JOB_DATE,
    JOB_COUNT;
```

Observation:

JOB Order Date	Vendor Name	Job Order Count	Total COST	
-----	-----	-----	-----	

31-2023-03	CANADIAN NATIONAL	4	1111411.0000	
28-2023-04	USA TRANSPORT INC.	9	1029683.8000	

Final Answer: The top 2 vendors with maximum cost within the given date range are:

1. CANADIAN NATIONAL with a total cost of 1,111,411.0 (on 2023-03-31)
2. USA TRANSPORT INC. with a total cost of 1,029,683.8 (on 2023-04-28)""

},

{

"input": "Shows top 2 customers with maximum cost within date range",

"output": ""Thought: I need the customer and their associated costs for the given date range to write the SQL query

Action: ask_for_parameter

Action Input: "startDate, endDate" Asking for the date range

Observation: I can now use this to query the database

Thought: I can now query the database to get the top 2 customers with maximum cost within the given date range

Action: sql_db_query

Action Input:

SELECT TOP 2 WITH TIES

FORMAT(IML_JOB_ORDER_HEADER.JOB_ORDER_DATE, 'yyyy-MM') AS JOB_DATE,

BKGBL_JO_INTERFACE_ADDRESS.NAME_OF_THE_CUSTOMER,

SUM(IML_JOB_ORDER_COST.JOB_ORDER_COST) AS COST

FROM

IML_JOB_ORDER_HEADER

JOIN

IML_JOB_ORDER_CONTAINER con

ON IML_JOB_ORDER_HEADER.JOB_ORDER_SEQ = con.JOB_ORDER_SEQ

```
JOIN
    BKGBL_JO_INTERFACE_ADDRESS
ON con.REF_NUM = BKGBL_JO_INTERFACE_ADDRESS.BKGBL_INTERFACE_NO
JOIN
    IML_JOB_ORDER_COST
ON con.JOB_ORDER_CONTAINER_SEQ = IML_JOB_ORDER_COST.JOB_ORDER_COST_SEQ
WHERE
    IML_JOB_ORDER_HEADER.JOB_ORDER_DATE >= '2023-01-01'
    AND IML_JOB_ORDER_HEADER.JOB_ORDER_DATE < '2024-01-01'
GROUP BY
    FORMAT(IML_JOB_ORDER_HEADER.JOB_ORDER_DATE, 'yyyy-MM'),
    BKGBL_JO_INTERFACE_ADDRESS.NAME_OF_THE_CUSTOMER
ORDER BY
    COST DESC;
```

Observation:

JOB Order Date	Customer Name	Total COST
-----	-----	-----
2023-07	CONTAINERPORT DOING DRAYAGE FROM	968529.02
2023-07	FIRST COAST LOGISTICS OF MIAMI	651778.66

Final Answer: The top 2 customers with maximum cost within the given date range are:

- 1. CONTAINERPORT DOING DRAYAGE FROM with a total cost of 968,529.02 (on 2023-07)
 - 2. FIRST COAST LOGISTICS OF MIAMI with a total cost of 651,778.66 (on 2023-07)"""
- },
- {

"input": "Top 2 routes with maximum cost within date range based on size and type",

"output": ""Thought: I need the equipment size, type, and the cost for the given date range to write the SQL query

Action: ask_for_parameter

Action Input: "startDate, endDate" Asking for the date range

Observation: I can now use this to query the database

Thought: I can now query the database to get the top 2 routes with maximum cost based on size and type within the given date range

Action: sql_db_query

Action Input:

```
SELECT TOP 2
    FORMAT(jh.JOB_ORDER_DATE, 'dd-yyyy-MM') AS JOB_DATE,
    ijc.EQ_SIZE,
    ijc.EQ_TYPE,
    frm_scope.ZONE_CODE,
    frm_scope.GEO_SCOPE_CODE AS FROM_LOCATION,
    to_scope.GEO_SCOPE_CODE AS TO_LOCATION,
    COUNT(*) AS JOB_COUNT,
    SUM(jh.JOB_ORDER_COST) AS COST
FROM IML_JOB_ORDER_HEADER jh
JOIN IML_JOB_ORDER_CONTAINER ijc
    ON jh.JOB_ORDER_SEQ = ijc.JOB_ORDER_SEQ
JOIN CAP_GEO_SCOPE frm_scope
    ON jh.JOB_ORDER_FRMLOC = frm_scope.GEO_SCOPE_SEQ
JOIN CAP_GEO_SCOPE to_scope
    ON jh.JOB_ORDER_TOLOC = to_scope.GEO_SCOPE_SEQ
WHERE jh.JOB_ORDER_DATE BETWEEN '2023-01-01' AND '2023-12-31'
```

AND frm_scope.ZONE_CODE = 'USNC'

GROUP BY

FORMAT(jh.JOB_ORDER_DATE, 'dd-yyyy-MM'),

ijc.EQ_SIZE,

ijc.EQ_TYPE,

frm_scope.ZONE_CODE,

frm_scope.GEO_SCOPE_CODE,

to_scope.GEO_SCOPE_CODE

ORDER BY

SUM(jh.JOB_ORDER_COST) DESC,

ijc.EQ_SIZE,

ijc.EQ_TYPE;

Observation:

Job Order Date	Equipment Size	Equipment Type	Zone Code	From Location	To Location	Job Order Count	Total Cost
----------------	----------------	----------------	-----------	---------------	-------------	-----------------	------------

19-2023-12	40	HD	USNC	USPOM	USPOM	28	2984480.80
------------	----	----	------	-------	-------	----	------------

08-2023-11	45	DW	USNC	DTCNC	DTCNC	25	2124524.00
------------	----	----	------	-------	-------	----	------------

Final Answer: The top 2 routes with the maximum cost within the date range based on equipment size and type are:

1. 40 HD Equipment from USPOM to USPOM with a total cost of 2,984,480.80 (on 2023-12-19)

2. 45 DW Equipment from DTCNC to DTCNC with a total cost of 2,124,524.00 (on 2023-11-08)"""

},

{

"input": "Get bookings that require spotting within the range between 1 April 2015 to 30 April 2016",

"output": ""Thought: I need the date range to write the SQL query

Action: ask_for_parameter

Action Input: "startDate" and "endDate"

Observation: I can now use these dates to parameterize the query.

Thought: I can now query the database

Action: sql_db_query

Action Input:

```
SELECT
    BKG_NO,
    CARGO_TYPE,
    VESSEL_CODE,
    VOYAGE_SML,
    POL,
    POL_TYPE,
    SPOT_LOC,
    SPOT_LOC_TYPE,
    CASE
        WHEN POL_TYPE = 'PT' THEN
            (SELECT port_name FROM cap_port WHERE port_code = POL)
        WHEN POL_TYPE IN ('DP', 'TM', 'RR') THEN
            (SELECT terminal_depot_name FROM cap_terminal_depot WHERE terminal_depot_code = POL)
        WHEN POL_TYPE = 'CT' THEN
            (SELECT city_name FROM cap_city WHERE city_code = POL)
        ELSE NULL
    END AS POL_NAME,
```

EQ_SIZE,
EQ_TYPE,
SAIL_DATE,
SAIL_TIME,
TRANS_MODE,
JO_STATUS,
ZONE_CODE,
SHIPPER_ADDRESS,
SPOT_DATE,
SPOT_TIME,
EQUIPMENT_NO
FROM (

SELECT DISTINCT

BKG.BKG_NO,
BKG.CARGO_TYPE,
BKG.VESSEL_CODE,
BKG.VOYAGE_SML,
CAP1.GEO_SCOPE_CODE AS POL,
CAP1.GEO_SCOPE_TYPE AS POL_TYPE,
CAP2.GEO_SCOPE_CODE AS SPOT_LOC,
CAP2.GEO_SCOPE_TYPE AS SPOT_LOC_TYPE,
BKG.EQ_SIZE,
BKG.EQ_TYPE,
BKG.SAIL_DATE,
BKG.SAIL_TIME,


```

    BKG.TRANS_MODE,
    " AS MT_SPOT_DATE,
    BKG.JO_STATUS,
    CAP1.STATE_CODE AS POL_STATE_CODE,
    CAP2.ZONE_CODE,
    BKG.SHIPPER_ADDRESS,
    BKG.SPOT_DATE,
    BKG.SPOT_TIME,
    BKG.EQUIPMENT_NO,
    BKG.EQUIPMENT_LINE_NO
FROM
    BKG_JO_INTERFACE BKG
    JOIN CAP_GEO_SCOPE CAP1 ON CAP1.GEO_SCOPE_SEQ = BKG.POL_TERMINAL_SEQ
    JOIN CAP_GEO_SCOPE CAP2 ON CAP2.GEO_SCOPE_SEQ = BKG.SHIPPER_CITY_SEQ
WHERE
    ISNULL(BKG.JO_STATUS, 'N') = 'N'
    AND CAST(BKG.SPOT_DATE AS DATE) BETWEEN '2015-04-01' AND '2016-04-30'
) AS T
ORDER BY SAIL_DATE;

```

Observation: The query fetches bookings within the specified date range.

Final Answer: The bookings within the given date range are shown.""""

},

{

"input": "Show me top 2 container size type with maximum cost within date range 1 Jan 2019 to 1 Dec 2019",

"output": ""Thought: I need the date range to write the SQL query

Action: ask_for_parameter

Action Input: "startDate" and "endDate"

Observation: I can now use these dates to parameterize the query.

Thought: I can now query the database

Action: sql_db_query

Action Input:

```
SELECT TOP 2
```

```
A.COUNTRY_CODE,
```

```
A.GLOBLE_ROUTE,
```

```
(SELECT GEO_SCOPE_CODE
```

```
FROM CAP_GEO_SCOPE
```

```
WHERE GEO_SCOPE_SEQ = A.FROM_LOC_VALUE) AS FROM_LOC_CODE,
```

```
(SELECT GEO_SCOPE_CODE
```

```
FROM CAP_GEO_SCOPE
```

```
WHERE GEO_SCOPE_SEQ = A.TO_LOC_VALUE) AS TO_LOC_CODE,
```

```
A.FROM_LOC_VALUE,
```

```
A.TO_LOC_VALUE,
```

```
CASE A.IMPORT_EXPORT
```

```
  WHEN 'I' THEN 'IMPORT'
```

```
  WHEN 'E' THEN 'EXPORT'
```

```
  ELSE A.IMPORT_EXPORT
```

END AS IMPORT_EXPORT,

CASE A.FROM_TYPE

WHEN 'ZN' THEN 'ZONE'

WHEN 'CO' THEN 'COUNTRY'

WHEN 'ST' THEN 'STATE'

WHEN 'CT' THEN 'CITY'

WHEN 'TM' THEN 'TERMINAL'

WHEN 'PT' THEN 'PORT'

WHEN 'RR' THEN 'RAIL RAMP'

ELSE A.FROM_TYPE

END AS FROM_TYPE,

CASE A.TO_TYPE

WHEN 'ZN' THEN 'ZONE'

WHEN 'CO' THEN 'COUNTRY'

WHEN 'ST' THEN 'STATE'

WHEN 'CT' THEN 'CITY'

WHEN 'TM' THEN 'TERMINAL'

WHEN 'PT' THEN 'PORT'

WHEN 'RR' THEN 'RAIL RAMP'

ELSE A.TO_TYPE

END AS TO_TYPE,

A.DRAYAGE_MARKUP1_BY,

A.DRAYAGE_MARKUP1,
A.DRAYAGE_MARKUP2,
A.RECORD_STATUS,
A.RECORD_ADD_USER,
A.RECORD_ADD_DATE,
A.RECORD_CHG_USER,
A.RECORD_CHG_DATE,
A.SHORT_HAUL_MARKUP1_BY,
A.SHORT_MARKUP_1,
A.SHORT_MARKUP_2,
A.LONG_HAUL_MARKUP1_BY,
A.LONG_MARKUP_1,
A.LONG_MARKUP_2,
A.RAIL_MARKUP1_BY,
A.RAIL_MARKUP_1,
A.RAIL_MARKUP_2,
A.DRAYAGE_MARKUP2_BY,
A.SHORT_MARKUP2_BY,
A.LONG_MARKUP2_BY,
A.RAIL_MARKUP2_BY,
A.REVISION_NO,
A.EFFECTIVE_DATE,
A.EXPIRY_DATE
FROM IML_MARKUP_SETUP_DETAILS A
WHERE CAST(A.EFFECTIVE_DATE AS DATE) BETWEEN '2019-01-01' AND '2019-12-01';

Observation: The query fetches the top 2 container size types with maximum cost in the given date range.

Final Answer: The top 2 container size types with maximum cost are displayed."""

},

{

"input": "Give me top 5 Job orders wise cost and margin",

"output": ""Thought: I need the top 5 job orders based on cost and margin

Action: I can now query the database

Action Input: select top 5

(select GEO_SCOPE_CODE from CAP_GEO_SCOPE where GEO_SCOPE_SEQ =
JOB_ORDER_FRMLOC) as From_location,

(select GEO_SCOPE_CODE from CAP_GEO_SCOPE where GEO_SCOPE_SEQ =
JOB_ORDER_TOLOC) as TO_location,

(select REGION_CODE from CAP_GEO_SCOPE where GEO_SCOPE_SEQ = JOB_ORDER_TOLOC) as
REGION_CODE,

(select COUNTRY_CODE from CAP_GEO_SCOPE where GEO_SCOPE_SEQ = JOB_ORDER_TOLOC)
as COUNTRY_CODE,

(select ZONE_CODE from CAP_GEO_SCOPE where GEO_SCOPE_SEQ = JOB_ORDER_TOLOC) as
ZONE_CODE,

(select CITY_CODE from CAP_GEO_SCOPE where GEO_SCOPE_SEQ = JOB_ORDER_TOLOC) as
CITY_CODE,

JOB_ORDER_NUM,

MAX(JOB_ORDER_COST) as JOB_ORDER_COST,

MAX(TARIFF_BASE_MRKUP2) as TARIFF_BASE_MRKUP2,

MAX(BASE_MAX_COST_MRKUP1) as BASE_MAX_COST_MRKUP1,

MAX(SALES) as SALES

from IML_JOB_ORDER_HEADER

group by JOB_ORDER_FRMLOC, JOB_ORDER_TOLOC, JOB_ORDER_NUM

order by 11 desc

Observation: Here are the top 5 job orders by cost and margin

Final Answer:

```
| From_location | TO_Location | REGION_CODE | COUNTRY_CODE | ZONE_CODE | CITY_CODE |  
JOB_ORDER_NUM | JOB_ORDER_COST | TARIFF_BASE_MRKUP2 | BASE_MAX_COST_MRKUP1 | SALES  
|
```

```
|-----|-----|-----|-----|-----|-----|-----|-----|-----  
-----|-----|-----|
```

```
| USPOM    | DTCNC    | NAM      | US      | USNC    | USCLT    | SML21000338A |  
18198000.0000 | 20017950    | 20017800    | 20017875 |
```

```
| DTCNC    | USPOM    | NAM      | US      | USFLL    | USMIA    | SML21000362A |  
17424000.0000 | 19166550    | 19166400    | 19166475 |
```

```
| DTCNC    | USPOM    | NAM      | US      | USFLL    | USMIA    | SML21000282A |  
18396000.0000 | 19166550    | 19166400    | 19166475 |
```

```
| DTMGA    | USPOM    | NAM      | US      | USFLL    | USMIA    | SML21000318A |  
12546000.0000 | 13800750    | 13800600    | 13800675 |
```

```
| USPOM    | USPOM    | NAM      | US      | USFLL    | USMIA    | MIA19111992A |  
167501.4200   | 1809690.732 | 988666.076   | 1399178.404 |
```

""""

}

,

{

"input": "Show me the repositioning request, status wise?",

"output": """"Thought: I need to classify the repositioning requests based on their status

Action: I can now query the database

Action Input: select

CASE

WHEN record_status = 'S' THEN 'Send For JO'

WHEN record_status = 'O' THEN 'Open'

```

        WHEN record_status = 'I' THEN 'Completed'

        ELSE 'Cancelled'

    END as Request_Status,

    Count(1) as Repositioning_Request_Status_Total

from EQT_REPOSITIONING h

group by RECORD_STATUS

```

Observation: Here are the repositioning requests according to their status

Final Answer:

```

+-----+-----+
| Request Status      | Repositioning Request Total |
+-----+-----+
| Cancelled           | 17                           |
| Open                | 42                           |
| Send For JO         | 812                          |
| Completed           | 1473                         |
+-----+-----+

```

"""

}

,

{

"input": "Who are the top 5 vendors with maximum job orders for March 2023 at export & import based on job order?",

"output": ""Thought: I need the top 5 vendors with the most job orders in March 2023.

Action: I can now query the database

Action Input: SELECT TOP 5 WITH TIES

```

    FORMAT(jh.JOB_ORDER_DATE, 'dd-yyyy-MM') AS JOB_DATE,

```

```

    bp.PARTNER_NAME,
    COUNT(*) AS JOB_COUNT
    --SUM(jh.JOB_ORDER_COST) AS COST
FROM IML_JOB_ORDER_HEADER jh
JOIN BPM_PARTNER_NAME bp
    ON jh.BIZ_PARTNER_SEQ = bp.BIZ_PARTNER_SEQ
WHERE
    bp.RECORD_STATUS = 'A'
    AND jh.JOB_ORDER_DATE >= '2023-03-01'
    AND jh.JOB_ORDER_DATE < '2023-03-31'
    AND JH.JOB_ORDER_STATUS NOT IN ('R','Z','C')
GROUP BY
    FORMAT(jh.JOB_ORDER_DATE, 'dd-yyyy-MM'),
    jh.BIZ_PARTNER_SEQ,
    bp.PARTNER_NAME
ORDER BY
    JOB_COUNT DESC;

```

Observation: Here are the list of vendors with maximum job orders for March 2023

Final Answer:

JOB_DATE	PARTNER_NAME	JOB_COUNT
06-2023-03	A & V EXPRESS INC	16
16-2023-03	ZNN XPRESS INC.	14
13-2023-03	TRUCKCONTRACT1MARCH12	13

| 13-2023-03 | TRUCKCONTRACT2MARCH12 | 13 |

| 13-2023-03 | ZNN XPRESS INC. | 11 |

+-----+-----+-----+

""

}

,

{

"input": "Give me the list of vendor with their proforma amount pending for invoice",

"output": ""Thought: I need the list of vendors with their pending proforma amounts

Action: I can now query the database

Action Input: SELECT C.PARTNER_NAME AS Vendor_Name,

SUM(I.PROFORMA_AMT - I.MATCHED_AMT) AS Proforma_Balance,

SUM(I.PROFORMA_AMT - I.MATCHED_AMT) AS To_Be_Invoice

FROM FPR_PROFORMA_ITEM I

INNER JOIN BPM_BIZ_PARTNER B ON I.VENDOR_CODE = B.PARTNER_CODE

INNER JOIN dbo.BPM_PARTNER_NAME C ON B.BIZ_PARTNER_SEQ = C.BIZ_PARTNER_SEQ

GROUP BY C.PARTNER_NAME

HAVING SUM(I.PROFORMA_AMT - I.MATCHED_AMT) > 0;

Observation: Here are the list of Vendor whose proforma amount are pending for Invoice

Final Answer:

+-----+-----+-----+

| Vendor_Name | Proforma_Balance | To_Be_Invoice |

+-----+-----+-----+

| 61 TRANSPORT DIVISION, EVANS DELIVERY CO | 166.7100 | 166.7100 |

| A & V EXPRESS INC | 475930.1600 | 475930.1600 |

AGS TRANSPORT, INC.	5821.6200	5821.6200	
AIM SOUTH LOGISTICS	2641.1800	2641.1800	
ALLEGIANT INTERMODAL (EVANS DELIVERY COMPANY)		7.2300	7.2300

+-----+-----+-----+

""""

}

]