Snowflake -Inventory Project

```
// Step to Categorize product // ABC ANALYSIS
-Find the annual revenue of each product
-Then divide it by total revenue get the percentage contribution of each product
- Sort the percentage in desc order and find the cumulative average
-Then sort the product based on cumulative average
        { CUMULATIVE_APG <=70 THEN 'A [HIGH VALUE] '
            CUMULATIVE_APG <=90 THEN 'B [MEDIUM VALUE]'
            CUMULATIVE_APG > 90 THEN 'C LOW VALUE'       }
```

Sales Table Schema

```
create or replace TABLE INVENTORY.PROJECT.SALES (
    ORDER_NUMBER VARCHAR(16777216),
    ORDER_DATE DATE,
    SKU_ID VARCHAR(10),
    WAREHOUSE_ID VARCHAR(10),
    CUSTOMER_TYPE VARCHAR(50),
    ORDER_QUANTITY NUMBER(38,0)
);
```

Inventory control table Schema

```
create or replace TABLE INVENTORY.PROJECT.INVENTORY_CONTROL (
    SKU_ID VARCHAR(20),
    VENDOR_NAME VARCHAR(255),
    WAREHOUSE_ID VARCHAR(20),
    CURRENT_INVENTORY_QUANTITY NUMBER(38,0),
    COST_PER_SKU FLOAT,
    TOTAL_VALUE NUMBER(15,2),
    UNITS VARCHAR(5),
    AVERAGE_LEAD_TIME NUMBER(38,0),
    MAXIMUM_LEAD_TIME NUMBER(38,0),
    UNIT_PRICE FLOAT
);
```

Annual Order Quantity

```
SELECT SKU_ID,SUM(ORDER_QUANTITY) AS ANNUAL_ORDER_QUANTITY
FROM SALES
WHERE ORDER_DATE >= DATEADD(YEAR,-1,CURRENT_DATE())
GROUP BY SKU_ID ORDER BY SKU_ID
```

ABC ANLAYSIS

```
CREATE VIEW ABC_ANALYSIS AS(

WITH CTE AS
(
    select ic.SKU_ID,ic.unit_price ,
    coalesce(aoq.annual_order_quantity,0) as Annual_OQT,
    ROUND(UNIT_PRICE*ANNUAL_OQT,2) AS ANNUAL_REVENUE,
    SUM(ANNUAL_REVENUE) OVER () AS TOTAL
    from inventory_control ic
    left join annual_order_qnty aoq on ic.sku_id = aoq.sku_id
```

```
    ORDER BY ANNUAL_REVENUE ASC
),

CTE2 AS
(

    SELECT * , round((annual_revenue/total)*100,2) as annual_pct,
    sum(annual_pct) over(order by annual_revenue desc) as cumulative_apg
    FROM CTE
)

SELECT SKU_ID,UNIT_PRICE,ANNUAL_REVENUE,
CASE
WHEN CUMULATIVE_APG <=70 THEN 'A [HIGH VALUE] '
WHEN CUMULATIVE_APG <=90 THEN 'B [MEDIUM VALUE]'
ELSE 'C LOW VALUE' END AS ABC

FROM CTE2
ORDER BY CUMULATIVE_APG) -- APG AVERAGE PERCENT
```

Dim Calendar table / For date references

```
CREATE OR REPLACE TABLE MY_DATE_DIMENSION (
   MY_DATE          DATE        NOT NULL
  ,YEAR             SMALLINT    NOT NULL
  ,MONTH            SMALLINT    NOT NULL
  ,MONTH_NAME       CHAR(3)     NOT NULL
  ,DAY_OF_MON       SMALLINT    NOT NULL
  ,DAY_OF_WEEK      VARCHAR(9)  NOT NULL
  ,WEEK_OF_YEAR     SMALLINT    NOT NULL
  ,DAY_OF_YEAR      SMALLINT    NOT NULL
)
AS
  WITH CTE_MY_DATE AS (
    SELECT DATEADD(DAY, SEQ4(), '2010-01-01') AS MY_DATE
      FROM TABLE(GENERATOR(ROWCOUNT=>5000))  -- Number of days after reference date in pr
  )
  SELECT MY_DATE
        ,YEAR(MY_DATE)
        ,MONTH(MY_DATE)
        ,MONTHNAME(MY_DATE)
        ,DAY(MY_DATE)
        ,DAYOFWEEK(MY_DATE)
        ,WEEKOFYEAR(MY_DATE)
        ,DAYOFYEAR(MY_DATE)
    FROM CTE_MY_DATE
;
```

Dim Product WEEKLY reference table for proper weekly average calculation view

```
create view dim_product_weekly as (
with CTE as

    (
    select  distinct YEAR, WEEK_OF_YEAR as week from  dim_date
    WHERE MY_DATE >= DATEADD(YEAR,-1,CURRENT_DATE()) and MY_DATE <= CURRENT_DATE()

    )

select * from cte
cross join sku s
order by sku_id , year , week
)
```

Weekly-wise product standard deviation and CV for products in order tables.

```
CREATE VIEW product_cf  as
WITH CTE AS
(
    select wp.*,coalesce(weekly_or_qty,0)as qnty
    from dim_product_weekly wp
    left join weekly_order wo on wp.sku_id = wo.sku_id and wp.week = wo.week and wp.year=
)

SELECT SKU_ID,SKU_NAME , AVG(QNTY) AS WEEKLY_AVERAGE , STDDEV(QNTY) AS STANDEV ,
STANDEV/WEEKLY_AVERAGE AS CF // NAMING ERROR CF --CV //
FROM CTE
GROUP BY SKU_ID,SKU_NAME
HAVING WEEKLY_AVERAGE > 0
```

Product ABC and XYZ Complete Full Analysis Final / All Tables Joined

```
CREATE VIEW  FULL_INENTORY_ABC_XYZ AS
WITH cte AS
(
   select ic.*,abc.annual_revenue,
    cf.weekly_average,cf.standev,coalesce (cf.cf,1000) as cv,
    ntile(10) over(order by cv) as rn,abc
    from inventory_control ic
    left join product_cf cf on cf.sku_id = ic.sku_id
    left join abc_analysis abc on abc.sku_id = ic.sku_id
)

select *,
CASE
WHEN rn <=2 THEN 'X [UNIFORM DEMAND]'
WHEN rn <= 5 THEN 'Y [VARIABLE DEMAND]'
ELSE 'Z [UNCERTAIN DEMAND]' END as  XYZ
from cte
```

Selecting sales data from the last year.**//IMPORTANT**

```
CREATE VIEW DIM_SALES_ANALYSIS AS
SELECT * FROM SALES
WHERE ORDER_DATE >= DATEADD(YEAR,-1,CURRENT_DATE())
```

Product sales and quantities from the sales table, corresponding to orders that are available each week. // Some products don't receive orders every week, so this needs to be considered with a join operation on date dimension table.

```
CREATE VIEW  SALES_DATA_GROUP_BY_WEEKLY AS
WITH CTE AS

(
    SELECT MAX(ORDER_DATE) AS DATES ,
    YEAR(ORDER_DATE) AS YEAR ,
    WEEK(ORDER_DATE) AS WEEK ,
    SKU_ID,SUM(ORDER_QUANTITY) AS QUANTITY

    FROM DIM_SALES_ANALYSIS -------------------------- VIEW
    GROUP BY YEAR, WEEK ,SKU_ID

)

SELECT C1.*,IC.UNIT_PRICE,IC.UNIT_PRICE * QUANTITY AS SALES_AMOUNT
FROM CTE C1
JOIN INVENTORY_CONTROL IC ON IC.SKU_ID = C1.SKU_ID
```

Weekly Sales Data for Each Product (All Weeks Considered) // TABLE USED FINAL FOR ANALYSIS

```
CREATE VIEW F_WEEKLY_DEMAND_EACH_PRODUCT AS

SELECT W.* ,
COALESCE(S.QUANTITY,0) AS QUANTITY,
S.UNIT_PRICE,
COALESCE(S.SALES_AMOUNT,0) AS SALES_AMOUNT

FROM WEEKLY_DIM_PRODUCT_WITH_DATES w ----------------------VIEW
LEFT JOIN SALES_DATA_GROUP_BY_WEEKLY s on w.sku_id = s.sku_id and
w.year =s.year and w.week=s.week
```

CALCULATION FORMULAS

```
# INVENTORY TURN OVER RATIO
INVENTORY TURN OVER RATIO = SUM(ANNUAL REVENUE)/SUM(TOTAL VALUE)

#SAFTEY STOCK
SAFTEY STOCK = PEAK DEMAND * MAX(LEAD TIME) - AVERAGE DEMAND * AVG( LEAD TIME )

# REORDER POINT
REORDER POINT =  SAFTEY STOCK + AVERAGE DEMAND * AVG( LEAD TIME )
```

Peak Demand for Each Product

```
CREATE VIEW PRODUCT_PEAK_DEMAND AS
select SKU_ID , MAX(QUANTITY) AS PEAK_DEMAND
from F_WEEKLY_DEMAND_EACH_PRODUCT
GROUP BY SKU_ID
```

This is the final table for our detailed stock analysis. The table includes vital information such as the ABC and XYZ classifications, which are essential tools for inventory control. It also provides calculated safety stock levels, designed to prevent stockouts caused by variability in supply and demand. The table also includes the reorder point information, which is the level of inventory at which a new order should be placed to replenish stock before it runs out. Lastly, the table presents the stock status, providing a comprehensive view of the current inventory situation.

```
//AVERAGE LEAD TIME and MAX LEAD TIME are given in monthly basics so need to convert
to weekly basis //

WITH CTE AS
(
    SELECT IC.*,PP.PEAK_DEMAND FROM FULL_INENTORY_ABC_XYZ IC
    JOIN PRODUCT_PEAK_DEMAND PP ON IC.SKU_ID = PP.SKU_ID
),

SAFTEY_STOCK AS
(
    SELECT * ,ROUND((PEAK_DEMAND * (MAXIMUM_LEAD_TIME/7)) - (WEEKLY_AVERAGE * (AVERAGE_LE
    FROM CTE
),

REORDER_POINT AS
(
    SELECT * , ROUND(SAFTEY_STOCK + (WEEKLY_AVERAGE * (AVERAGE_LEAD_TIME/7)),0) AS REORDE
    FROM SAFTEY_STOCK
)

SELECT * ,CURRENT_INVENTORY_QUANTITY AS CURRENT_AVAILABLE_STOCK ,
CASE
WHEN CURRENT_INVENTORY_QUANTITY = 0 THEN 'YES'
WHEN CURRENT_INVENTORY_QUANTITY > SAFTEY_STOCK THEN 'NO'
ELSE 'YES'  END AS TO_ORDER,

CASE
WHEN CURRENT_INVENTORY_QUANTITY = 0 THEN 'OUT OF STOCK '
WHEN CURRENT_INVENTORY_QUANTITY < REORDER_POINT  THEN 'BELOW REORDER '
ELSE 'IN STOCK ' END AS STOCK_STATUS

FROM REORDER_POINT
```