KARACHI BRANCH INVENTORY VALUE (LATEST TP)

QUERY-1

Calculate inventory value for Karachi branch items using the latest cost price (TP). Final Output Fields: branch_name, sku, item_name, unit_qty, total_inventory_value_tp.

```
WITH LatestCostPrice AS (
    -- Finds the latest cost price for each item via max receiving_item.id for
relevant branches
    SELECT
        ri.cost_price,
        ri.pos_id,
        i.sku
    FROM
        receiving_item ri
    JOIN
        item i
        ON ri.pos_item_id = i.pos_item_id
        AND ri.pos_id = i.pos_id
    JOIN
        pos p
        ON p.id = i.pos_id
        AND p.branch_name LIKE '%Kara%'
    WHERE
        ri.id IN (
            SELECT MAX(ri_inner.id)
            FROM receiving_item ri_inner
            JOIN item i_inner
                ON ri_inner.pos_item_id = i_inner.pos_item_id
                AND ri_inner.pos_id = i_inner.pos_id
            JOIN receiving r_inner
                ON r_inner.id = ri_inner.receiving_id
            WHERE
                ri_inner.is_deleted = FALSE
                AND ri_inner.is_active = TRUE
                AND i_inner.is_deleted = FALSE
                AND i_inner.is_active = TRUE
                AND r_inner.is_deleted = FALSE
                AND r_inner.is_active = TRUE
            GROUP BY
                i_inner.pos_id,
                i_inner.pos_item_id
        )
),
CurrentStock AS (
    -- Gets current stock for active items in Karachi branches, excluding
test/invalid SKUs
    SELECT
```

```
        i.pos_id,
        p.branch_name,
        i.sku,
        i.name AS item_name,
        iq.quantity AS unit_qty
    FROM
        item i
    JOIN
        item_quantity iq
        ON iq.pos_item_id = i.pos_item_id
        AND i.pos_id = iq.pos_id
    JOIN
        pos p
        ON p.id = i.pos_id
    WHERE
        i.is_active = TRUE
        AND i.is_deleted = FALSE
        AND iq.is_deleted = FALSE
        AND iq.quantity > 0
        AND p.pos_is_active = TRUE
        AND p.branch_name LIKE '%Kara%'
        AND i.strip_size != '0'
        AND i.pack_size != '0'
        AND i.sku NOT LIKE '%zero%'
        AND i.sku NOT IN (
            'FORTES_6555',
            'FORTES_22759',
            'MULTIP_22674',
            'DULOSO_22673',
            '17747q',
            'TEST12_21931',
            'FORTES_21932')
        AND i.sku NOT LIKE '%e%'
        AND i.sku NOT LIKE '%t%')
```

Final Calculation

```
SELECT
    cs.branch_name,
    cs.sku,
    cs.item_name,
    cs.unit_qty,
    SUM(COALESCE(lcp.cost_price, 0) * cs.unit_qty) AS total_inventory_value_tp
FROM
    CurrentStock cs
LEFT JOIN
    LatestCostPrice lcp
    ON cs.pos_id = lcp.pos_id AND cs.sku = lcp.sku
WHERE
    cs.branch_name LIKE '%Karac%'  -- Final branch name check
```

```
GROUP BY
    cs.branch_name, cs.sku, cs.item_name, cs.unit_qty
ORDER BY
    cs.branch_name, cs.item_name;
```

BRANCH-WISE MONTHLY GMV (USING MRP)

QUERY-2

Calculate monthly Gross Merchandise Value (GMV) per branch/POS using item MRP. Final Output Fields: sale_year, sale_month, branch, pos_id, monthly_gmv_mrp

```
WITH PreparedSalesData AS (
 -- Prepares sales data, adjusts timezone, applies filters
    SELECT
        (sale.sale_time - interval '4 hour') AS sale_time_local, -- Verify TZ
adjustment
        sale.pos_id,
        pos.branch_name as branch,
        sale_item.quantity_purchased,
        sale_item.item_mrp
    FROM
        sale_item
    JOIN
        sale ON sale.id = sale_item.sale_id
    JOIN
        pos ON sale.pos_id = pos.id
```

```
    WHERE
        sale.pos_sale_id NOT IN (1150, 1889, 2336, 2598, 3713) -- Exclude
specific sales
        AND (sale.customer_id NOT IN (33, 37, 69, 71, 55, 57) OR
sale.customer_id IS NULL) -- Exclude internal customers
        AND sale.is_deleted = FALSE
        AND DATE(sale.sale_time - interval '4 hour') >= '2023-01-01' -- Date
filter
        AND sale_item.discount < 99 -- Exclude high discounts)
```

Final Aggregation

```
SELECT
    EXTRACT(YEAR FROM sale_time_local) as sale_year,
    EXTRACT(MONTH FROM sale_time_local) as sale_month,
    branch,
    pos_id,
    SUM(COALESCE(quantity_purchased, 0) * COALESCE(item_mrp, 0)) as
monthly_gmv_mrp
FROM
    PreparedSalesData
GROUP BY
    sale_year,
    sale_month,
    branch,
    pos_id
ORDER BY
    sale_year,
    sale_month,
    branch,
    pos_id;
```

ITEM LEVEL INVENTORY (SPECIFIC POS)

QUERY-3

Retrieve inventory quantity and value for POS ID 9 using the highest recorded MRP.

Final Output Fields: sku, product_name, pos_item_id, pos_id, highest_recorded_mrp,available_quantity, total_inventory_value_mrp

```
WITH CurrentStockAtPOS9 AS (
  --Fetches active items with quantity > 0 for POS ID 9
  SELECT
    i.sku,
    i.name AS product_name,
    i.pos_item_id,
    i.pos_id,
    iq.quantity
  FROM
    item i
  JOIN
    item_quantity iq
    ON iq.pos_item_id = i.pos_item_id
    AND iq.pos_id = i.pos_id
  WHERE
    i.pos_id = 9
    AND iq.is_deleted = FALSE
    AND iq.is_active = TRUE
    AND i.is_deleted = FALSE
    AND i.is_active = TRUE
    AND iq.quantity > 0
),
MaxMRPAtPOS9 AS (
```

Finds the maximum recorded MRP for each item at POS ID 9

```
MaxMRPAtPOS9 AS (
    SELECT
        rib.pos_item_id,
        rib.pos_id,
        MAX(rib.item_mrp) AS max_item_mrp
    FROM
        receiving_item_batch rib
    WHERE
        rib.pos_id = 9
        AND rib.is_deleted = FALSE
        AND rib.is_active = TRUE
        AND rib.item_mrp IS NOT NULL
    GROUP BY
        rib.pos_item_id,
        rib.pos_id)
```

Final Calculation

```
SELECT
  cs.sku,
```

```
    cs.product_name,
    cs.pos_item_id,
    cs.pos_id,
    COALESCE(mm.max_item_mrp, 0) AS highest_recorded_mrp,
    cs.quantity AS available_quantity,
    (COALESCE(mm.max_item_mrp, 0) * cs.quantity) AS total_inventory_value_mrp
FROM
    CurrentStockAtPOS9 cs
LEFT JOIN
    MaxMRPAtPOS9 mm
    ON cs.pos_item_id = mm.pos_item_id
    AND cs.pos_id = mm.pos_id
ORDER BY
    cs.product_name;
```

DAILY RECEIVING SUMMARY (FILTERED BRANCHES/INVOICE)

QUERY-4

Summarize items received yesterday for branches 3-8 with invoice containing 'MMT'. Final Output Fields: receiving_date_local, branch_name, ..., receiving_day

```
WITH DailyReceivingData AS (
  -- Aggregates receiving data for specified POS IDs, adjusts timestamp
  SELECT
    DATE(r.created_at - interval '4 hour') AS receiving_date_local, -- Verify TZ adjustment
    p.branch_name,
    r.po_invoice_number,
    ri.pos_item_id,
    i.sku,
    i.name AS item_name,
    SUM(ri.total_quantity) AS total_quantity_received,
    SUM(ri.quantity_purchased) AS quantity_purchased,
    SUM(ri.quantity_return) AS quantity_returned
  FROM
    receiving r
  LEFT JOIN
    receiving_item ri ON ri.receiving_id = r.id
  LEFT JOIN
    item i ON i.pos_item_id = ri.pos_item_id AND i.pos_id = ri.pos_id
  LEFT JOIN
    pos p ON r.pos_id = p.id
  WHERE
    r.pos_id IN (3, 4, 5, 6, 7, 8) -- Specific branches
    AND r.is_active = TRUE
    AND r.is_deleted = FALSE
    AND ri.is_active = TRUE
    AND ri.is_deleted = FALSE
    AND i.is_active = TRUE
    AND i.is_deleted = FALSE
    AND p.pos_is_active = TRUE
  GROUP BY
    DATE(r.created_at - interval '4 hour'),
    p.branch_name,
    r.po_invoice_number,
    ri.pos_item_id,
    i.sku,
    i.name)
```

Final Selection

```
SELECT
  drd.*,
  EXTRACT(MONTH FROM drd.receiving_date_local) AS receiving_month,
  EXTRACT(YEAR FROM drd.receiving_date_local) AS receiving_year,
  EXTRACT(DAY FROM drd.receiving_date_local) AS receiving_day
```

```
FROM
  DailyReceivingData drd
WHERE
  drd.receiving_date_local = (CURRENT_DATE - interval '1 day') -- Yesterday's data
  AND drd.po_invoice_number LIKE '%MMT%' -- Specific invoice pattern
ORDER BY
  drd.branch_name,
  drd.receiving_date_local,
  drd.item_name;
```

IB2C SKU LEVEL ORDER REPORT

QUERY-5

Detailed SKU-level report for Institutional B2C orders including quantities and pricing. Final Output Fields: order_date_pkt, Order_ID_Reference, ..., Sub_Total.

```
WITH BaseOrderData AS (
  -- Gathers core order, item, batch details, converting timestamp to PKT
  SELECT
    o.id AS order_id,
    DATE_FORMAT(date(CONVERT_TZ(o.created_at,'+00:00','+05:00')), '%m/%d/%y') AS
order_date_pkt,
    o.reference_id,
    o.daraz_order_id,
    o.consignment_number,
    io.pin,
    io.invoice_no,
    CASE o.hub_id
      WHEN 5293 THEN 'Karachi'
      WHEN 14317 THEN 'Islamabad'
      WHEN 17317 THEN 'Lahore'
      ELSE 'Unknown'
    END AS warehouse_location,
    CASE
      WHEN i.name IS NOT NULL THEN i.name
      WHEN o.customer_id = '25612' THEN 'OMI Hospital'
      ELSE 'B2C'
    END AS institution_name,
    os.name AS order_status_name,
    iof.description AS institution_offer_desc,
    p.b2c_product_display_name AS product_sku_name,
    oib.order_item_id,
    oib.quantity AS shipped_quantity_batch,
    (oib.quantity * oib.actual_price) AS batch_total_before_discount,
    (oib.quantity * oib.selling_price) AS batch_total_after_discount
  FROM
    orders o
  JOIN
    order_item oi ON o.id = oi.order_id
  JOIN
    products p ON oi.products_id = p.id
  JOIN
    order_items_batch oib ON oib.order_item_id = oi.id
  JOIN
    order_status os ON o.status = os.id
  LEFT JOIN
    institutional_orders io ON o.id = io.order_id
  LEFT JOIN
    institutions i ON io.institution_id = i.id
  LEFT JOIN
```

```
    institutional_offers iof ON io.institutional_offer_id = iof.id
  WHERE
    o.business_type IN ('IB2C')
```

Replace placeholders or use variables for dates

```
AND date(CONVERT_TZ(o.created_at,'+00:00','+05:00')) BETWEEN '2023-01-01' AND '2023-12-
31' -- Example Date Range
  AND o.contact_person_name NOT LIKE '%Test%'
  AND o.customer_id NOT IN (370, 456258, 52930, 1293333)
  AND o.contact_person_name NOT LIKE '%window%'
  AND oib.is_deleted = 0
),
AggregatedBatchData AS (
```

Summarizes shipped quantities and values per order item

```
SELECT
    order_item_id,
    SUM(shipped_quantity_batch) AS total_shipped_quantity,
    SUM(batch_total_before_discount) AS total_value_before_discount,
    SUM(batch_total_after_discount) AS total_value_after_discount
FROM
    BaseOrderData
GROUP BY
    order_item_id;
```