SQL ASSIGNMENT 2

Mixed Party + Order Queries

5.1 Shipping Addresses for October 2023 Orders

Business Problem:
Customer Service might need to verify addresses for orders placed or completed in October 2023. This helps ensure shipments are delivered correctly and prevents address-related issues.

Fields to Retrieve:

- `ORDER_ID`
- `PARTY_ID` (Customer ID)
- `CUSTOMER_NAME` (or FIRST_NAME / LAST_NAME)
- `STREET_ADDRESS`
- `CITY`
- `STATE_PROVINCE`
- `POSTAL_CODE`
- `COUNTRY_CODE`
- `ORDER_STATUS`
- `ORDER_DATE`

`Solution:`

```
SELECT

oh.order_id,

pcm.party_id,

per.first_name,

per.last_name,

pad.address1,

pad.address2,

pad.city,

pad.state_province_geo_id,
```

```
pad.postal_code,

pad.country_geo_id,

oh.Entry_date

FROM order_header oh

JOIN order_contact_mech ocm ON ocm.order_id=oh.order_id AND
ocm.CONTACT_MECH_PURPOSE_TYPE_ID='SHIPPING_LOCATION'

JOIN party_contact_mech pcm ON pcm.contact_mech_id=ocm.contact_mech_id

JOIN postal_address pad ON pad.contact_mech_id=ocm.contact_mech_id

JOIN person per ON per.party_id=pcm.party_id

WHERE date(oh.Entry_date)>='2023-10-01' AND
date(oh.Entry_date)<='2023-11-01';
```

5.2 Orders from New York

Business Problem:
Companies often want region-specific analysis to plan local marketing, staffing, or promotions in certain areas—here, specifically, New York.

Fields to Retrieve:

- `ORDER_ID`
- `CUSTOMER_NAME`
- `STREET_ADDRESS` (or shipping address detail)
- `CITY`
- `STATE_PROVINCE`
- `POSTAL_CODE`
- `TOTAL_AMOUNT`
- `ORDER_DATE`
- `ORDER_STATUS`

Solution:

```
SELECT
        oh.ORDER_ID ,
        p.FIRST_NAME ||p.LAST_NAME AS CUSTOMER_NAME,
        pa.ADDRESS2,
        pa.CITY,
        pa.STATE_PROVINCE_GEO_ID,
        pa.POSTAL_CODE,
        oh.GRAND_TOTAL,
        oh.ORDER_DATE,
        oh.STATUS_ID
        FROM
        Person p JOIN  Party_CONTACT_MECH pcm
        on
        p.party_id=pcm.party_id
        LEFT JOIN
        Postal_Address pa on pcm.contact_mech_id= pa.contact_mech_id
        LEFT JOIN
        ORDER_CONTACT_MECH ocm ON ocm.ORDER_ID=pcm.CONTACT_MECH_ID
        JOIN
        ORDER_HEADER oh ON ocm.ORDER_ID=oh.ORDER_ID
        WHERE pa.CITY='NEW YORK';
```

5.3 Top-Selling Product in New York

Business Problem:
Merchandising teams need to identify the best-selling product(s) in a specific region (New York) for targeted restocking or promotions.

Fields to Retrieve:

- `PRODUCT_ID`
- `INTERNAL_NAME`
- `TOTAL_QUANTITY_SOLD`
- `CITY` / `STATE` (within New York region)
- `REVENUE` (optionally, total sales amount)

Solution:

```
SELECT
p.product_id,
p.INTERNAL_NAME,
oi.QUANTITY as TOTAL_QUANTITY_SOLD,
pa.CITY,
oh.GRAND_TOTAL AS REVENUE
FROM
ORDER_ITEM oi JOIN ORDER_HEADER oh ON oi.ORDER_ID=oh.ORDER_ID
JOIN
ORDER_CONTACT_MECH ocm ON oh.ORDER_ID=ocm.ORDER_ID
JOIN
POSTAL_ADDRESS pa ON ocm.CONTACT_MECH_ID=pa.CONTACT_MECH_ID
JOIN Product p On oi.PRODUCT_ID=p.PRODUCT_ID
WHERE
pa.CITY='NEW YORK' AND oh.STATUS_ID='ORDER_COMPLETED'
ORDER BY
REVENUE DESC;
```

7.3 Store-Specific (Facility-Wise) Revenue

Business Problem:
Different physical or online stores (facilities) may have varying levels of performance. The business wants to compare revenue across facilities for sales planning and budgeting.

Fields to Retrieve:

- FACILITY_ID
- FACILITY_NAME
- TOTAL_ORDERS
- TOTAL_REVENUE
- DATE_RANGE

Solution:

```
SELECT
        f.FACILITY_ID,
        f.FACILITY_NAME,
        COUNT( DISTINCT oh.ORDER_ID) AS TOTAL_ORDERS,
        SUM(oi.QUANTITY * oi.UNIT_PRICE) AS TOTAL_REVENUE
        MIN(oh.ORDER_DATE) || ' to ' || MAX(oh.ORDER_DATE) AS DATE_RANGE
```

```
FROM
	FACILITY f
JOIN
	ORDER_HEADER oh ON f.FACILITY_ID = oh.ORIGIN_FACILITY_ID
JOIN
	ORDER_ITEM oi ON oh.ORDER_ID = oi.ORDER_ID
WHERE
	oh.STATUS_ID IN ('ORDER_COMPLETED', 'ORDER_SHIPPED')  -- Filtering only relevant orders
GROUP BY
	f.FACILITY_ID, f.FACILITY_NAME
ORDER BY
	TOTAL_REVENUE DESC;
```

8. Inventory Management & Transfers

8.1 Lost and Damaged Inventory

Business Problem:
Warehouse managers need to track “shrinkage” such as lost or damaged inventory to reconcile physical vs. system counts.

Fields to Retrieve:

- `INVENTORY_ITEM_ID`
- `PRODUCT_ID`
- `FACILITY_ID`
- `QUANTITY_LOST_OR_DAMAGED`
- `REASON_CODE` (Lost, Damaged, Expired, etc.)
- `TRANSACTION_DATE`

`Solution:`

```
SELECT p.product_id,

ii.inventory_item_id,

iiv.variance_reason_id,

ii.facility_id,
```

```
count(p.product_id) as product_total

FROM product p JOIN inventory_item ii ON ii.PRODUCT_ID=p.PRODUCT_ID

JOIN inventory_item_variance iiv ON
iiv.INVENTORY_ITEM_ID=ii.INVENTORY_ITEM_ID

WHERE iiv.variance_reason_id is not null and
iiv.variance_reason_id='VAR_LOST' OR
iiv.variance_reason_id='VAR_DAMAGED'

Group By
ii.facility_id,p.product_id,iiv.variance_reason_id,ii.inventory_item_i
d;
```

8.2 Low Stock or Out of Stock Items Report

Business Problem:
Avoiding out-of-stock situations is critical. This report flags items that have fallen below a certain reorder threshold or have zero available stock.

Fields to Retrieve:

- `PRODUCT_ID`
- `PRODUCT_NAME`
- `FACILITY_ID`
- `QOH` (Quantity on Hand)
- `ATP` (Available to Promise)
- `REORDER_THRESHOLD`
- `DATE_CHECKED`

`Solution:`

```
SELECT
    ii.product_id,
    p.product_Name,
    ii.QUANTITY_ON_HAND_TOTAL AS QOH,
    ii.AVAILABLE_TO_PROMISE_TOTAL AS ATP,
    pf.MINIMUM_STOCK AS REORDER_QUANTITY
    FROM
    INVENTORY_ITEM ii
```

```
        JOIN Product p ON
        ii.product_id=p.product_id
        JOIN PRODUCT_FACILITY pf ON
        p.product_id= pf.product_id
        WHERE
        ii.QUANTITY_ON_HAND_TOTAL <= pf.REORDER_QUANTITY
        OR ii.QUANTITY_ON_HAND_TOTAL = 0
ORDER BY
        ii.QUANTITY_ON_HAND_TOTAL ASC;
```

8.3 Retrieve the Current Facility (Physical or Virtual) of Open Orders

Business Problem:
The business wants to know where open orders are currently assigned, whether in a physical store or a virtual facility (e.g., a distribution center or online fulfillment location).

Fields to Retrieve:

- ORDER_ID
- ORDER_STATUS
- FACILITY_ID
- FACILITY_NAME
- FACILITY_TYPE_ID

Solution:

```
SELECT oh.order_id,
        oh.status_id AS order_status,
        ii.facility_id ,
        f.facility_name,
        f.facility_type_id
        FROM order_header oh
        JOIN inventory_item_detail iid on
        iid.order_id=oh.order_id
        JOIN inventory_item  ii on
        ii.inventory_item_id=iid.inventory_item_id
        JOIN
        Facility f on f.facility_id=ii.facility_id
        where
        oh.status_id  IN ('ORDER_APPROVED','ORDER_HOLD') ;
```

8.4 Items Where QOH and ATP Differ

Business Problem:
Sometimes the **Quantity on Hand (QOH)** doesn't match the **Available to Promise (ATP)** due to pending orders, reservations, or data discrepancies. This needs review for accurate fulfillment planning.

Fields to Retrieve:

- `PRODUCT_ID`
- `FACILITY_ID`
- `QOH` (Quantity on Hand)
- `ATP` (Available to Promise)
- `DIFFERENCE` (QOH - ATP)

`Solution:`

```
SELECT

    ii.product_id,

    ii.facility_id,

    ii.quantityOnHandTotal as QOH,

    ii.availableToPromise as ATP,

    ii.qunatityOnHandTotal -ii.availableTopromise as DIFFERENCE

    From

    InventoryItem ii

    WHERE (ii.quantityOnHandTotal -ii.availableToPromise) <> 0

    ORDER  BY  DIFFERENCE DESC;
```

8.5 Order Item Current Status Changed Date-Time

Business Problem:
Operations teams need to audit when an order item's status (e.g., from "Pending" to "Shipped") was last changed, for shipment tracking or dispute resolution.

Fields to Retrieve:

- ORDER_ID
- ORDER_ITEM_SEQ_ID
- CURRENT_STATUS_ID
- STATUS_CHANGE_DATETIME
- CHANGED_BY

Solution:

```
SELECT
        oi.ORDER_ID,
        oi.ORDER_ITEM_SEQ_ID,
        oi.Status_id As CURRENT_STATUS_ID,
        oi.LAST_UPDATED_STAMP AS STATUS_CHANGE_DATETIME,
        oi.Change_BY_USER_LOGIN_ID AS CHANGED_BY
        From ORDER_ITEM oi
        ORDER BY oi.ORDER_ID;
```

8.6 Total Orders by Sales Channel

Business Problem:
Marketing and sales teams want to see how many orders come from each channel (e.g., web, mobile app, in-store POS, marketplace) to allocate resources effectively.

Fields to Retrieve:

- SALES_CHANNEL
- TOTAL_ORDERS
- TOTAL_REVENUE
- REPORTING_PERIOD

Solution:

```
SELECT
oh.sales_Channel_Enum_id As Channel,
Count(oh.order_id) As Total_Orders,
SUM(oi.UNIT_PRICE *oi.Quantity)AS TOTAL_REVENUE,
Date(oh.order_date)
FROM
ORDER_HEADER oh
JOIN
ORDER_ITEM oi ON oh.order_id=oi.order_id
GROUP BY oh.sales_Channel_Enum_Id,DATE(oh.order_date);
```