SQL Queries (Data Loading in Snowflake)-:

1. **Creation of Warehouse and Database-:**

```
--Creating a loading warehouse
CREATE OR REPLACE WAREHOUSE loading_wh WITH
WAREHOUSE_SIZE='X-SMALL'
AUTO_RESUME = TRUE -- default
AUTO_SUSPEND = 600 -- default
INITIALLY_SUSPENDED = TRUE; -- default

--Create a new database named Fetch
CREATE DATABASE IF NOT EXSITS FETCH;

--Setting the Context
USE fetch.public;
USE WAREHOUSE loading_wh;
```

2. **User Table Creation -:**

```
--Creating the users table
CREATE OR REPLACE TABLE users(v variant);

--Creating users_cleaned table
CREATE OR REPLACE TABLE FETCH.PUBLIC.USERS_CLEANED (
   user_id STRING,
   is_active BOOLEAN,
   created_date TIMESTAMP,
   last_login TIMESTAMP,
   user_role STRING,
   sign_up_source STRING,
   user_state STRING
);

--Inserting data to users_cleaned
INSERT INTO FETCH.PUBLIC.USERS_CLEANED
SELECT
   V:"_id":"$oid"::STRING AS user_id,
   V:"active"::BOOLEAN AS is_active,
   TO_TIMESTAMP(V:"createdDate":"$date"::NUMBER / 1000) AS created_date,
   TO_TIMESTAMP(V:"lastLogin":"$date"::NUMBER / 1000) AS last_login,
   V:"role"::STRING AS user_role,
   V:"signUpSource"::STRING AS sign_up_source,
   V:"state"::STRING AS user_state
FROM FETCH.PUBLIC.USERS;

SELECT COUNT(*) FROM FETCH.PUBLIC.USERS_CLEANED;
```

```
--Removing the repetition of the User data table

CREATE OR REPLACE TABLE users_cleaned AS

SELECT * FROM (
   SELECT *,
        ROW_NUMBER() OVER (PARTITION BY user_id ORDER BY created_date DESC) AS rn
   FROM users_cleaned
) WHERE rn = 1;
```

3. Brands Table Creation-:

```
--Creating the brands table
CREATE OR REPLACE TABLE FETCH.PUBLIC.BRANDS(v VARIANT);

SELECT * FROM FETCH.PUBLIC.BRANDS LIMIT 5;

--Reading sample data from brands dataset
SELECT
   V:"_id":"$oid"::STRING AS brand_id,
   V:"barcode"::STRING AS barcode,
   V:"brandCode"::STRING AS brand_code,
   V:"category"::STRING AS category,
   V:"categoryCode"::STRING AS category_code,
   V:"cpg":"$id":"$oid"::STRING AS cpg_id,
   V:"cpg":"$ref"::STRING AS cpg_ref,
   V:"name"::STRING AS name,
   V:"topBrand"::BOOLEAN AS is_top_brand
FROM FETCH.PUBLIC.BRANDS
LIMIT 10;

--Creating brands_cleaned table
CREATE OR REPLACE TABLE FETCH.PUBLIC.BRANDS_CLEANED (
   brand_id STRING,
   barcode STRING,
   brand_code STRING,
   category STRING,
   category_code STRING,
   cpg_id STRING,
   cpg_ref STRING,
   name STRING,
   is_top_brand BOOLEAN
);


--Inserting into brands_cleaned table
```

```
INSERT INTO FETCH.PUBLIC.BRANDS_CLEANED
SELECT
   V:"_id":"$oid"::STRING,
   V:"barcode"::STRING,
   V:"brandCode"::STRING,
   V:"category"::STRING,
   V:"categoryCode"::STRING,
   V:"cpg":"$id":"$oid"::STRING,
   V:"cpg":"$ref"::STRING,
   V:"name"::STRING,
   V:"topBrand"::BOOLEAN
FROM FETCH.PUBLIC.BRANDS;

SELECT * FROM FETCH.PUBLIC.BRANDS_CLEANED LIMIT 5;
```

4. Receipts (Receipts_cleaned and Receipt_items) Tables Creation-:

```
--Creating the Receipts table
CREATE OR REPLACE TABLE FETCH.PUBLIC.RECEIPTS(v VARIANT);

--Reading data from receipts dataset
SELECT
   v:"_id":"$oid"::STRING AS receipt_id,
   v:"bonusPointsEarned"::INT AS bonus_points,
   v:"bonusPointsEarnedReason"::STRING AS bonus_reason,
   TO_TIMESTAMP(v:"createDate":"$date"::NUMBER/1000) AS create_date,
   TO_TIMESTAMP(v:"dateScanned":"$date"::NUMBER/1000) AS date_scanned,
   TO_TIMESTAMP(v:"finishedDate":"$date"::NUMBER/1000) AS finished_date,
   TO_TIMESTAMP(v:"modifyDate":"$date"::NUMBER/1000) AS modify_date,
   TO_TIMESTAMP(v:"pointsAwardedDate":"$date"::NUMBER/1000) AS points_awarded_date,
   v:"pointsEarned"::FLOAT AS points_earned,
   TO_TIMESTAMP(v:"purchaseDate":"$date"::NUMBER/1000) AS purchase_date,
   v:"purchasedItemCount"::INT AS purchased_item_count,
   v:"rewardsReceiptStatus"::STRING AS receipt_status,
   v:"totalSpent"::FLOAT AS total_spent,
   v:"userId"::STRING AS user_id
FROM FETCH.PUBLIC.RECEIPTS
LIMIT 10;

--Creating receipts_cleaned table for DML
CREATE OR REPLACE TABLE FETCH.PUBLIC.RECEIPTS_CLEANED (
   receipt_id STRING,
   bonus_points INT,
   bonus_reason STRING,
   create_date TIMESTAMP,
   date_scanned TIMESTAMP,
   finished_date TIMESTAMP,
   modify_date TIMESTAMP,
   points_awarded_date TIMESTAMP,
```

```
    points_earned FLOAT,
    purchase_date TIMESTAMP,
    purchased_item_count INT,
    receipt_status STRING,
    total_spent FLOAT,
    user_id STRING
);

--Inserting to receipts_cleaned data set
INSERT INTO FETCH.PUBLIC.RECEIPTS_CLEANED
SELECT
    v:"_id":"$oid"::STRING,
    v:"bonusPointsEarned"::INT,
    v:"bonusPointsEarnedReason"::STRING,
    TO_TIMESTAMP(v:"createDate":"$date"::NUMBER/1000) AS create_date,
    TO_TIMESTAMP(v:"dateScanned":"$date"::NUMBER/1000) AS date_scanned,
    TO_TIMESTAMP(v:"finishedDate":"$date"::NUMBER/1000) AS finished_date,
    TO_TIMESTAMP(v:"modifyDate":"$date"::NUMBER/1000) AS modify_date,
    TO_TIMESTAMP(v:"pointsAwardedDate":"$date"::NUMBER/1000) AS points_awarded_date,
    v:"pointsEarned"::FLOAT,
    TO_TIMESTAMP(v:"purchaseDate":"$date"::NUMBER/1000) AS purchase_date,
    v:"purchasedItemCount"::INT,
    v:"rewardsReceiptStatus"::STRING,
    v:"totalSpent"::FLOAT,
    v:"userId"::STRING
FROM FETCH.PUBLIC.RECEIPTS;

SELECT * FROM FETCH.PUBLIC.RECEIPTS_CLEANED;

SELECT
    v:"_id":"$oid"::STRING AS receipt_id,
    v:"userId"::STRING AS user_id,
    v:"purchaseDate":"$date"::TIMESTAMP AS purchase_date,
    v:"totalSpent"::FLOAT AS total_spent,
    v:"rewardsReceiptStatus"::STRING AS receipt_status,
    -- Flattened rewardsReceiptItemList
    f.value:"barcode"::STRING AS item_barcode,
    f.value:"description"::STRING AS item_description,
    f.value:"itemPrice"::FLOAT AS item_price,
    f.value:"finalPrice"::FLOAT AS final_price,
    f.value:"quantityPurchased"::INT AS quantity_purchased,
    f.value:"partnerItemId"::STRING AS partner_item_id,
    -- User Flags
    f.value:"userFlaggedBarcode"::STRING AS user_flagged_barcode,
    f.value:"userFlaggedDescription"::STRING AS user_flagged_description,
    f.value:"userFlaggedPrice"::FLOAT AS user_flagged_price,
    f.value:"userFlaggedQuantity"::INT AS user_flagged_quantity,
    f.value:"userFlaggedNewItem"::BOOLEAN AS user_flagged_new_item,
    -- Review Flags
```

```
    f.value:"needsFetchReview"::BOOLEAN AS needs_fetch_review,
    f.value:"needsFetchReviewReason"::STRING AS fetch_review_reason,
    -- Points & Rewards
    f.value:"pointsEarned"::FLOAT AS points_earned,
    f.value:"pointsPayerId"::STRING AS points_payer_id,
    f.value:"rewardsGroup"::STRING AS rewards_group,
    f.value:"rewardsProductPartnerId"::STRING AS rewards_product_partner_id,
    -- Other Fields
    f.value:"preventTargetGapPoints"::BOOLEAN AS prevent_target_gap_points,
    f.value:"originalMetaBriteBarcode"::STRING AS original_meta_brite_barcode,
    f.value:"originalMetaBriteDescription"::STRING AS original_meta_brite_description
FROM FETCH.PUBLIC.RECEIPTS,
LATERAL FLATTEN(input => v:"rewardsReceiptItemList") f
LIMIT 20;

--Creating receipt_items table
CREATE OR REPLACE TABLE FETCH.PUBLIC.RECEIPT_ITEMS (
    receipt_id STRING,
    user_id STRING,
    purchase_date TIMESTAMP,
    total_spent FLOAT,
    receipt_status STRING,
    -- Item Details
    item_barcode STRING,
    item_description STRING,
    item_price FLOAT,
    final_price FLOAT,
    quantity_purchased INT,
    partner_item_id STRING,
    -- User Flags
    user_flagged_barcode STRING,
    user_flagged_description STRING,
    user_flagged_price FLOAT,
    user_flagged_quantity INT,
    user_flagged_new_item BOOLEAN,
    -- Review Flags
    needs_fetch_review BOOLEAN,
    fetch_review_reason STRING,
    -- Points & Rewards
    points_earned FLOAT,
    points_payer_id STRING,
    rewards_group STRING,
    rewards_product_partner_id STRING,
    -- Other Fields
    prevent_target_gap_points BOOLEAN,
    original_meta_brite_barcode STRING,
    original_meta_brite_description STRING
);
--Insert into receipt_items table
```

```
INSERT INTO FETCH.PUBLIC.RECEIPT_ITEMS
SELECT
   v:"_id":"$oid"::STRING,
   v:"userId"::STRING,
   TO_TIMESTAMP(V:"purchaseDate":"$date"::NUMBER / 1000) AS purchase_date,
   v:"totalSpent"::FLOAT,
   v:"rewardsReceiptStatus"::STRING,
   -- Flattened rewardsReceiptItemList
   f.value:"barcode"::STRING,
   f.value:"description"::STRING,
   f.value:"itemPrice"::FLOAT,
   f.value:"finalPrice"::FLOAT,
   f.value:"quantityPurchased"::INT,
   f.value:"partnerItemId"::STRING,
   -- User Flags
   f.value:"userFlaggedBarcode"::STRING,
   f.value:"userFlaggedDescription"::STRING,
   f.value:"userFlaggedPrice"::FLOAT,
   f.value:"userFlaggedQuantity"::INT,
   f.value:"userFlaggedNewItem"::BOOLEAN,
   -- Review Flags
   f.value:"needsFetchReview"::BOOLEAN,
   f.value:"needsFetchReviewReason"::STRING,
   -- Points & Rewards
   f.value:"pointsEarned"::FLOAT,
   f.value:"pointsPayerId"::STRING,
   f.value:"rewardsGroup"::STRING,
   f.value:"rewardsProductPartnerId"::STRING,
   -- Other Fields
   f.value:"preventTargetGapPoints"::BOOLEAN,
   f.value:"originalMetaBriteBarcode"::STRING,
   f.value:"originalMetaBriteDescription"::STRING
FROM FETCH.PUBLIC.RECEIPTS,
LATERAL FLATTEN(input => v:"rewardsReceiptItemList") f;

SELECT * FROM FETCH.PUBLIC.RECEIPT_ITEMS;
SELECT COUNT(*) FROM FETCH.PUBLIC.RECEIPT_ITEMS;
SELECT * FROM FETCH.PUBLIC.RECEIPTS_CLEANED;
```