SQL Queries for creating the tables

Stores table queries

```
INSERT INTO stores (store_id, store_name, store_city, store_region, opening_date)
SELECT DISTINCT
   store_id,
   store_name,
   store_city,
   store_region,
   NULL::DATE
FROM raw_clv_data
ON CONFLICT (store_id) DO NOTHING;
```

Adding Dates (to the null value at opening dates using first transaction date)

```
UPDATE stores s
SET opening_date = sub.first_tx
FROM (
   SELECT store_id, MIN(transaction_date)::DATE AS first_tx
   FROM retail_raw
   GROUP BY store_id
) sub
WHERE s.store_id = sub.store_id;
```

Products Table

```
INSERT INTO products (
   product_id,
   product_name,
   product_category,
   unit_price
)
SELECT DISTINCT
   product_id,
   product_name,
   product_category,
   unit_price
FROM raw_clv_data
ON CONFLICT (product_id) DO NOTHING;
```

Customer_details Table

```
INSERT INTO customer_details (
  customer_id,
  first_name,
  email,
  loyalty_status,
  total_loyalty_points,
  last_purchase_date,
  segment_id,
  customer_phone,
  customer_since
)
SELECT DISTINCT
  customer_id,
  first_name,
  email,
  loyalty_status,
  NULL::INT,    -- will compute later if needed
  NULL::DATE,   -- we will update properly next
  NULL,
  customer_phone,
  customer_since
FROM raw_clv_data
ON CONFLICT (customer_id) DO NOTHING;
```

Purchase Date

```
UPDATE customer_details c
SET last_purchase_date = sub.last_tx
FROM (
   SELECT customer_id, MAX(transaction_date)::DATE AS last_tx
   FROM retail_raw
   GROUP BY customer_id
) sub
WHERE c.customer_id = sub.customer_id;
```

Promotions Details

```
INSERT INTO promotion_details (
   promotion_id,
   promotion_name,
   start_date,
   end_date,
   discount_percentage,
   applicable_category
)
SELECT DISTINCT
   promotion_id,
   promotion_name,
   NULL::DATE,
   NULL::DATE,
   discount_percentage,
   NULL
FROM raw_clv_data
```

```
WHERE promotion_id IS NOT NULL
ON CONFLICT (promotion_id) DO NOTHING;
```

Promotion start and end dates

```
UPDATE promotion_details p
SET
  start_date = sub.first_seen,
  end_date = sub.last_seen
FROM (
  SELECT
    promotion_id,
    MIN(transaction_date)::DATE AS first_seen,
    MAX(transaction_date)::DATE AS last_seen
  FROM retail_raw
  WHERE promotion_id IS NOT NULL
  GROUP BY promotion_id
) sub
WHERE p.promotion_id = sub.promotion_id;
```

Store sales header

```
INSERT INTO store_sales_header (
  transaction_id,
  customer_id,
  store_id,
  transaction_date,
  total_amount,
  customer_phone
)
SELECT DISTINCT
  transaction_id,
  customer_id,
  store_id,
  transaction_date,
  final_amount,
  customer_phone
FROM raw_clv_data
ON CONFLICT (transaction_id) DO NOTHING;
```

Store sales line value

```
INSERT INTO store_sales_line_items (
   line_item_id,
   transaction_id,
   product_id,
   promotion_id,
   quantity,
   line_item_amount
)
SELECT
   ROW_NUMBER() OVER () AS line_item_id,
   transaction_id,
   product_id,
   promotion_id,
   quantity,
   final_amount
FROM raw_clv_data;
```