Sales Data Mart

# **Epic:** Build Sales Data Mart

This document contains detailed stories for creating the sales data mart for enabling Business Reporting.

**Data Source Files are located here:** <https://drive.google.com/drive/folders/1cfVJx6IicqLNKwWUIJG-O-htpTK3R-tE>

**Here’s the Link to Data Flow Mapping:** <https://drive.google.com/file/d/138nhSC686340IP8-6ijt8EP4SPd1l_1y/view?usp=drive_link>

## **Story 1:** Build **dim\_customers** table in **analytics.sales\_mart**

* **Description**As a Data Engineer  
  I want to ingest **Customers.xls**, cleanse it and publish conformed **dim\_customers** table in **analytics.sales\_mart**So that analysts can reliably join customer attributes to various fact tables.
* **Sub Tasks & Technical Specs**

1. **Ingest Raw file as a table in database**

* **Key Output:** customers table in **analytics.raw**
* **DDL**

| CREATE TABLE analytics.raw.customers(  customer\_id INT,  first\_name TEXT,  last\_name TEXT,  age INT,  country TEXT ); |
| --- |

* **Transformations**
  + All column names should be in lower case
  + Rename “First” to first\_name
  + Rename “Last” to last\_name
  + Data Types should match above
* **Acceptance Criteria**
  + All column names should be in lower case
  + Column names & Data Types should match with Key output
  + Total number of rows in the table should match with total rows in Customer.xls

1. **Build Staging Table with clean data**

* **Key Output: stg\_customers** table in **analytics.stage** based on the transformed data from **analytics.raw.customers**
* **DDL**

| CREATE TABLE analytics.stage.stg\_customers(  customer\_id INT,  first\_name TEXT,  last\_name TEXT,  age INT,  country TEXT ); |
| --- |

* **Transformations**
  + Remove numerics and special characters from **first\_name** & **last\_name** and add this transformed data into **analytics.raw.stg\_customers**

| INSERT INTO analytics.stage.stg\_customers SELECT   customer\_id,  TRANSLATE(first\_name, '1@!0', 'iaio') AS first\_name,  TRANSLATE(last\_name, '0', 'o') AS last\_name,  age,  country FROM raw.customers ; |
| --- |

* **Acceptance Criteria**
  + **first\_name** & **last\_name** should not contain any numerics and special characters.

1. **Build Staging Table with clean data**

* **Key Output: dim\_customers** table based on the data from analytics.stage.stg\_customers.
* **DDL**

| CREATE TABLE analytics.sales\_mart.dim\_customers (  customer\_ID INT PRIMARY KEY,  first\_name TEXT NOT NULL,  last\_name TEXT,  age INT NOT NULL,  country TEXT NOT NULL ); |
| --- |

* **Transformations**
  + Inserting data into **analytics.sales\_mart.dim\_customers**

| INSERT INTO analytics.sales\_mart.dim\_customers SELECT \*  FROM analytics.stage.stg\_customers WHERE customer\_id IS NOT NULL AND first\_name IS NOT NULL AND age IS NOT NULL AND country IS NOT NULL; |
| --- |

* **Acceptance Criteria**
  + One row per **customer\_id** (**PK**)
  + All **NOT NULL** columns enforced except **last\_name**
  + Age should non negative, non zero and less than 120

## **Story 2:** Build **fact\_orders** table in **analytics.sales\_mart**

* **Description**As a Data Engineer  
  I want to ingest **Order.csv**, cleanse it and publish conformed **fact\_orders** table in **analytics.sales\_mart**So that analysts can reliably join it with **dim\_customers** & slice/dice orders data.
* **Sub Tasks & Technical Specs**

1. **Ingest Raw file as a table in database**

* **Key Output:** orders table in **analytics.raw** schema
* **DDL**

| CREATE TABLE analytics.raw.orders (  order\_id INT,  item TEXT,  amount INT,  customer\_id INT ); |
| --- |

* **Transformations**
  + All column names should be in lower case
* **Acceptance Criteria**
  + Column names should be in lower case
  + Total number of rows in the table should match with total rows in Order.csv

1. **Build Staging Table with clean data**

* **Key Output: stg\_orders** table in **analytics.stage** schema.
* **DDL**

| CREATE TABLE analytics.stage.stg\_orders (  order\_id INT,  item TEXT,  amount INT,  customer\_id INT ); |
| --- |

* **Transformations**
  + No transformations are required as data has no data quality issues.
  + Load data from **analytics.raw.orders**

| INSERT INTO analytics.stage.stg\_orders SELECT \*  FROM analytics.raw.orders ; |
| --- |

* **Acceptance Criteria**
  + **analytics.stage.stg\_orders** should have data

1. **Build Staging Table with clean data**

* **Key Output: fact\_orders** table in **analytics.sales\_mart** schema
* **DDL**

| CREATE TABLE analytics.sales\_mart.fact\_orders(  order\_id INT PRIMARY KEY,  item TEXT NOT NULL,  amount INT NOT NULL,  customer\_id INT NOT NULL REFERENCES analytics.sales\_mart.dim\_customers(customer\_id) ); |
| --- |

* **Transformations**

| INSERT INTO analytics.sales\_mart.fact\_orders SELECT \*  FROM analytics.stage.stg\_orders WHERE order\_id IS NOT NULL AND item IS NOT NULL AND amount IS NOT NULL AND customer\_id IS NOT NULL  AND amount > 0  ; |
| --- |

* **Acceptance Criteria**
  + One row per **order\_id (PK)**
  + **NOT NULL** constraint is enforced on all of the columns
  + Amount should be greater than 0

## **Story 3:** Build **dim\_shipping\_status** table in **analytics.sales\_mart**

* **Description**As a Data Engineer  
  I want to ingest **Shipping.json**, cleanse it and publish conformed **dim\_shipping\_status** table in **analytics.sales\_mart**So that analysts can reliably join it with **dim\_customers** to generate shipping related reports.
* **Sub Tasks & Technical Specs**

1. **Ingest Raw file as a table in database**

* **Key Output: shipping\_json** table in **analytics.raw schema**
* **DDL**

| CREATE TABLE analytics.raw.shipping\_json (  data JSON ); |
| --- |

* **Transformations**
  + The **shipping.json** file contains an array of json objects. Convert it into a single column table that has one row per array item.
* **Acceptance Criteria**
  + Number of rows in the table should be equal to the number of items in the array

1. **Build Staging Table with clean data**

* **Key Output: stg\_shippings** table in **analytics.stage** schema
* **DDL**

| CREATE TABLE analytics.stage.stg\_shippings(  shipping\_id INT,  status TEXT,  customer\_id INT ); |
| --- |

* **Transformations**
  + Create a flat table from JSON
  + Remove duplicate combinations of **customer\_id** & **status.** This is based on the assumption that in shipping table a customer can only have either Delivered or Pending status if for a customer we have more than one value or Delivered/Pending then that would be considered as duplicate

| INSERT INTO analytics.stage.stg\_shippings WITH shippings AS (  SELECT   CAST(data->>'Shipping\_ID' AS INT) AS shipping\_id,  data->>'Status' AS status,  CAST(data->>'Customer\_ID' AS INT) AS customer\_id,  ROW\_NUMBER() OVER(PARTITION BY CAST(data->>'Customer\_ID' AS INT), data->>'Status') AS rn  FROM analytics.raw.shipping\_json ) SELECT  shipping\_id,  status,  customer\_id FROM shippings WHERE rn = 1 ; |
| --- |

* **Acceptance Criteria**
  + Table should have 3 columns: shipping\_id, status, customer\_id
  + The table should not have duplicates for combinations of **customer\_id** & **status.**

1. **Build Staging Table with clean data**

* **Key Output: dim\_shipping\_status** table in **analytics.sales\_mart** schema
* **DDL**

| CREATE TABLE analytics.sales\_mart.dim\_shipping\_status(  shipping\_id INT PRIMARY KEY,  status TEXT NOT NULL,  customer\_id INT NOT NULL REFERENCES analytics.sales\_mart.dim\_customers(customer\_id) ); |
| --- |

* **Transformations**
  + Inserting data into the dim\_shipping\_status

| INSERT INTO analytics.sales\_mart.dim\_shipping\_status SELECT \*  FROM analytics.stage.stg\_shippings WHERE shipping\_id IS NOT NULL AND status IS NOT NULL AND customer\_id IS NOT NULL; |
| --- |

* **Acceptance Criteria**
  + One row per **shipping\_id (PK)**
  + **NOT NULL** constraint is enforced on all of the columns
  + One customer can’t have duplicate status values

## Story 4: Create materialised views to enable Business Reporting

### BR 1: The total amount spent and the country for the Pending delivery status for each country.

**View Name:** mv\_country\_pending\_delivery\_spend

**SQL CODE:**

| SELECT   c.country,  SUM(o.amount) AS total\_amount FROM sales\_mart.dim\_customers c INNER JOIN sales\_mart.dim\_shipping\_status s ON c.customer\_id = s.customer\_id -- Inner join becasue not all customer have shipping data INNER JOIN sales\_mart.fact\_orders o ON c.customer\_id = o.customer\_id -- Inner join becasue not all customer have orders data WHERE s.status = 'Pending' -- to get customers with pending orders GROUP BY 1 ORDER BY 2 DESC ; |
| --- |

### BR 2: The total number of transactions, total quantity sold, and total amount spent for each customer, along with the product details.

**View Name:** mv\_customer\_product\_sales\_summary

**Assumptions:**

1. One order can be considered as one transaction as we don't have transaction level date
2. We don't have quantity sold column, for quantity i'll look at number of times an item is ordered and if it's ordered once

then order quantity would be 1 if it's ordered 10 times then quantity would be 10

**SQL CODE:**

| SELECT   c.customer\_id,  c.first\_name,  c.last\_name,  o.item,  COUNT(o.order\_id) AS total\_transactions,  COUNT(o.order\_id) AS total\_quantity,  SUM(o.amount) AS total\_amount\_spent FROM sales\_mart.dim\_customers c INNER JOIN sales\_mart.fact\_orders o ON c.customer\_id = o.customer\_id GROUP BY 1,2,3,4 ORDER BY 5 DESC ; |
| --- |

### BR 3: The maximum product purchased for each country.

**View Name:** mv\_max\_product\_per\_country

**SQL CODE:**

| WITH item\_quantity\_per\_country AS (  SELECT c.country,  o.item,  COUNT(o.order\_id) AS item\_sold\_cnt,  ROW\_NUMBER() OVER(PARTITION BY c.country ORDER BY COUNT(o.order\_id) DESC) AS rn  FROM sales\_mart.dim\_customers c  INNER JOIN sales\_mart.fact\_orders o ON c.customer\_id = o.customer\_id  GROUP BY 1,2 ) SELECT   country,  item,  item\_sold\_cnt FROM item\_quantity\_per\_country WHERE rn = 1 ; |
| --- |

### BR 4: The most purchased product based on the age category less than 30 and above 30.

**View Name:** mv\_top\_product\_by\_age\_category

**SQL CODE:**

| WITH items\_sold\_per\_age\_group AS (  SELECT   CASE WHEN c.age < 30 THEN 'Less than 30'  WHEN c.age >= 30 THEN '30 and Above'  END AS age\_range,  o.item,  COUNT(o.order\_id) AS item\_sold\_cnt  FROM sales\_mart.dim\_customers c  INNER JOIN sales\_mart.fact\_orders o ON c.customer\_id = o.customer\_id  GROUP BY 1,2 ), ranking AS (  SELECT  \*,  ROW\_NUMBER() OVER(PARTITION BY age\_range ORDER BY item\_sold\_cnt DESC) AS rn  FROM items\_sold\_per\_age\_group ) SELECT   age\_range,  item,  item\_sold\_cnt FROM ranking WHERE rn = 1 ; |
| --- |

### BR 5: The country that had minimum transactions and sales amount

**View Name:** mv\_country\_min\_sales\_and\_txn

**Assumptions:**

1. We don't have transactions table so I'll consider transaction amount = sales\_amount

**SQL CODE:**

| SELECT   c.country,  SUM(o.amount) AS sales\_amount FROM sales\_mart.dim\_customers c INNER JOIN sales\_mart.fact\_orders o ON c.customer\_id = o.customer\_id GROUP BY 1 ORDER BY 2 LIMIT 1 ; |
| --- |