**As a Data Analyst, you are required to**

* **Verify the accuracy, completeness, and reliability of source data. Show your results in a SQL or Python output.**

**describe customer;**

5 columns with NULL and DEFAULT constraints enabled.

**SELECT min(age) AS min\_age, max(age) AS max\_age, avg(age) AS avg\_age FROM customer;**

min\_age : 18

max\_age : 80

avg\_age : 47.5760

**# checking if customer\_id has duplicates or is null**

**select customer\_id from customer**

**group by customer\_id**

**having count(customer\_id) >1 or customer\_id is null;**

unique values with no nulls

select \* from customer

where age is null

or first is null

or last is null

or country is null

**# Count the number of customers from each country**

SELECT country, COUNT(\*) AS customer\_count FROM Customer GROUP BY country;

**Describe orders**

4 columns with NULL and DEFAULT constraint enabled.

**Checking for duplicates**

select order\_id from orders

group by order\_id

having count(order\_id) >1 or order\_id is null;

SELECT customer\_id, COUNT(\*) AS order\_count FROM Orders GROUP BY customer\_id;

**Checking Nulls**

select \* from orders

where Order\_ID is null

or item is null

– no null item and order\_id

SELECT status, COUNT(\*) AS status\_count FROM Shipping GROUP BY status;

SELECT customer\_id, COUNT(\*) AS shipment\_count FROM Shipping GROUP BY customer\_id;

SELECT \* FROM Shipping WHERE status = 'Pending';

**Checking accuracy of data:**

select count(c.customer\_id) from customer c

left join orders o on

c.customer\_id =o.customer\_id

left join shipping s on

c.customer\_id =s.customer\_id

where s.customer\_id is null

and order\_id is not null

**#94 customers have orders but shipping details are missing.**

select \* from customer c

left join orders o on

c.customer\_id =o.customer\_id

left join shipping s on

c.customer\_id =s.customer\_id

where s.customer\_id is null

and order\_id is null

**#35 customers don't have orders**

select status,count(shipping\_id) as cnt\_of\_orders from shipping

group by status

**pending: 150**

**Delivered : 100**

purchase cost

select avg(amount) as avg\_order\_amt,min(amount) AS lowest\_order\_amt,max(amount) AS maximum\_order\_amt from orders

**avg\_order\_amt : 2130.0000**

**lowest\_order\_amt : 200**

**maximum\_order\_amt : 12000**

**–Based on your findings, define and outline the requirements for anticipated datasets, detailing the necessary data components.**

1. Shipping table has missing data, depending on order id.

2. Order id should be the foreign key of the shipping table.

3. No way to calculate the number of items sold per order. (create new product table with product price) or order table should have quantity column

-- **Develop the data models to effectively organise and structure the information and provide a detailed mapping of existing data flows, focussing on the areas of concern.**

We should have a snowflake schema for the data model, rather than the star schema which is being used now.

**Areas of concern** :

• Product table is missing.

• In orders table,

• **order status** column should be there : if it is pending, cancelled, fulfilled, returned or delivered.

• Order date column should be there.

• In the shipping table, the order\_id column should be there to know what order is shipped out of multiple orders placed by the customer.

**Prepare a story with technical specifications for one part of the data model for a data engineer to include technical specifications and transformations. This story should give enough information for a Data Engineer to build table(s) and for a QA engineer to test it.**

**1. Business Context & Goal**

This story focuses on establishing the foundational data model for customer order fulfillment. The goal is to provide a unified and consistent view of customer information, their associated orders, and the shipping details for those orders. This data will be used for analytical reporting, customer segmentation, order tracking, and future data science initiatives.

**2. High-Level Requirements**

**Ingest**, **cleanse**, and **standardize** raw customer, order, and shipping data. **Establish relationships** and ensure **data quality** and **integrity** to provide a **robust**, **scalable data model** for downstream consumption.

**3.1. Source Systems**

Customer Data: CRM\_DB.Customers (PostgreSQL) - Daily full load from CRM.

Order Data: ECOM\_DB.Orders (MySQL) - Daily incremental load based on last\_updated\_timestamp.

Shipping Data: SHIPPING\_API.Shipments (REST API) - Daily batch pull.

**3.2. Target Database & Schema**

Database: Analytics\_Warehouse (Snowflake)

Schema: DWH\_CORE

**3.3. Table Specifications & Transformations**

The following tables will be created in the DWH\_CORE schema.

**Table : Shipping**

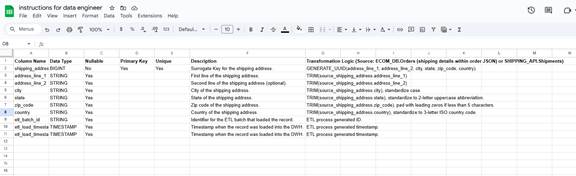
This table will contain distinct shipping addresses associated with orders.

**Purpose**: To normalize and store unique shipping address details, linked to orders.

**Type:** Dimension Table

**Update Strategy:** Type 0 (Fixed) or Type 1 (Overwriting) based on unique address key. If an address changes, it's considered a new address.

**Granularity:** One row per unique combination of address attributes.



**Quality Analyst job:**

When a data engineer creates a shipping\_table, QA engineers shift their focus to **Data Quality Testing** (or ETL Testing). This involves verifying the **quality, integrity, and reliability** of the data and the processes that populate the table.

**Data Quality Checks**

QA engineers perform comprehensive data quality checks:

* **Accuracy:** Validating data against source systems (**source-to-target validation**), verifying **calculated fields** (e.g., shipping\_duration), and ensuring adherence to **business rules** (e.g., shipping\_cost cannot be negative).
* **Completeness:** Checking for **nulls** in critical fields, verifying **row counts** against expectations, and identifying **missing data patterns**.
* **Consistency:** Confirming correct **data types** and **formats**, validating **referential integrity** with related tables, and ensuring **cross-table consistency** (e.g., shipping\_status aligning with order\_status).
* **Uniqueness:** Ensuring **primary keys** and other unique constraints have no duplicate values.
* **Timeliness/Freshness:** Verifying data **latency** and ensuring it meets business needs for reporting and analytics.
* **Validity:** Performing **domain checks** (e.g., shipping\_status values are within an allowed set) and **range checks** for numeric fields.

**ETL Process Testing**

Testing also focuses on how data enters the table via the data pipeline:

* **Source Data Extraction:** Verifying all expected data is extracted and handling missing or malformed source data.
* **Data Transformation Logic:** Ensuring correct **mapping** from source to target, testing all **transformation rules** (cleansing, aggregation), and evaluating **error handling**.
* **Data Loading:** Testing **incremental** and **full loads**, including **load performance**.
* **Idempotency:** Confirming consistent output even if the pipeline runs multiple times with the same input.
* **Schema Evolution:** How the pipeline handles changes in source or target schemas.

**Other Key Testing Areas**

* **Performance Testing:** Assessing **query performance** for downstream users and **load performance** for data ingestion.
* **Security Testing:** Verifying **access control** and proper **data masking/encryption** for sensitive information.
* **Monitoring and Alerting:** Ensuring mechanisms for **data quality monitoring** and **alerts** for issues or failures are in place.

**Methods and Tools**

QA engineers use a combination of:

* **SQL Queries** for direct data validation.
* **Scripting** (e.g., Python) for automation and comparison.
* **Data Quality Tools** (e.g., Great Expectations, dbt).
* **ETL Testing Tools** and **Data Profiling Tools**.

**Business Reporting Requirements**

These reporting requirements should be enabled based on the Data Analyst tasks above. The task is not to create the reports but to enable it.

*#1. the total amount spent and the country for the Pending delivery status for each country.*

*select country,sum(amount) as total\_amount\_spent from customer c*

*inner join orders o on o.customer\_id=c.customer\_id*

*inner join shipping s on s.customer\_id=c.customer\_id*

*where status='pending'*

*group by country;*

*UK 136300*

*USA 65500*

*UAE 53800*

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

*select concat(first, ' ',last) as name ,count(distinct order\_id) as no\_of\_orders,count(item) quantity\_sold,*

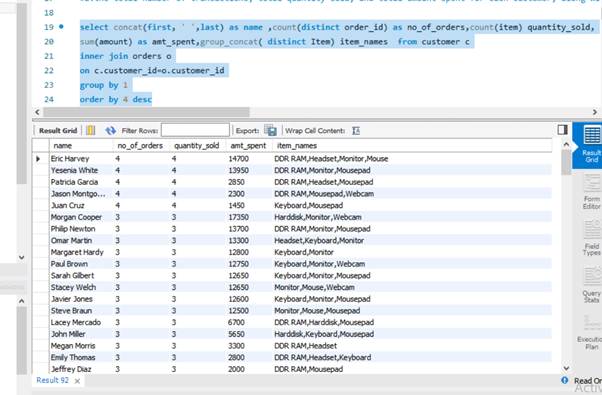
*sum(amount) as amt\_spent,group\_concat( distinct Item) item\_names from customer c*

*inner join orders o*

*on c.customer\_id=o.customer\_id*

*group by 1*

*order by 4 desc*

**

*#3.the maximum product purchased for each country.*

*with product\_purchased as*

*(select item,country,count(item) items\_purchased,*

*row\_number() over(partition by country order by count(item) desc,sum(amount) desc)as rank\_no*

*from customer c*

*inner join orders o*

*on o.customer\_id=c.customer\_id*

*group by 1,2*

*)*

*select item,country from product\_purchased*

*where rank\_no=1*

| *Keyboard* | *UAE* |
| --- | --- |
| *Mousepad* | *UK* |
| *Mousepad* | *USA* |

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

*select \* from*

*(select item as less\_than\_30*

*from orders o*

*inner join customer c*

*on o.customer\_id =c.customer\_id*

*group by item*

*order by count(case when age<30 then item end) desc*

*limit 1) a*

*cross join*

*(select item as more\_than\_30*

*from orders o*

*inner join customer c*

*on o.customer\_id =c.customer\_id*

*group by item*

*order by count(case when age>30 then item end) desc*

*limit 1) b*

*# less\_than\_30, more\_than\_30*

*'Mousepad', 'Keyboard'*

*#5.the country that had minimum transactions and sales amount.*

*with minimum\_transaction as*

*(select country,count(order\_id) as transactions,sum(amount) as sales\_amount,*

*dense\_rank() over( order by count(order\_id),sum(amount)) as min\_trans*

*from customer c*

*inner join orders o*

*on o.customer\_id=c.customer\_id*

*group by country*

*)*

*select country from minimum\_transaction*

*where min\_trans =1*

*Country*

*UAE*