Rational Database Design Report

1. Business Background

Superstore Background:

- Superstore operates as an e-commerce platform, offering a broad spectrum of product categories throughout the United States.
- With the steady growth in their business volume, their current data storage approach is proving insufficient.

Superstore Demands:

- As data volumes surge, they require a more efficient storage and retrieval system.
- They also need user-friendly mechanisms for daily operations and in-depth analysis.

Project Objectives:

- Construct a relational database for Superstore that:
  - Adheres to 3NF standards
  - Preserves the original data
  - Streamlines the data structure
  - Facilitates flexible querying

Original Data Structure:

- Format: Stored in an Excel spreadsheet.
- Organization: All data is amalgamated within a single sheet.
- Row Significance: Each row denotes a distinct item within a particular order.
- Data Arrangement: Columns are diversified to describe attributes like product details, order date, and customer information.

2. Creating ER Model

Entities:

- Within the original data, there are 5 distinct information clusters.
- In the relational database, these clusters can be represented by 5 entities:
  - Product Entity: Product details
  - Order Entity: Order date
  - Customer Entity: Order placer
  - Location Entity: Shipping address
  - Order_item Entity: Order amount

Assumption and Keys:

- Product to Order_item Relationship:
  - A Product may belong to one or more Orders_items; Each Order_item must have one and only one Product.

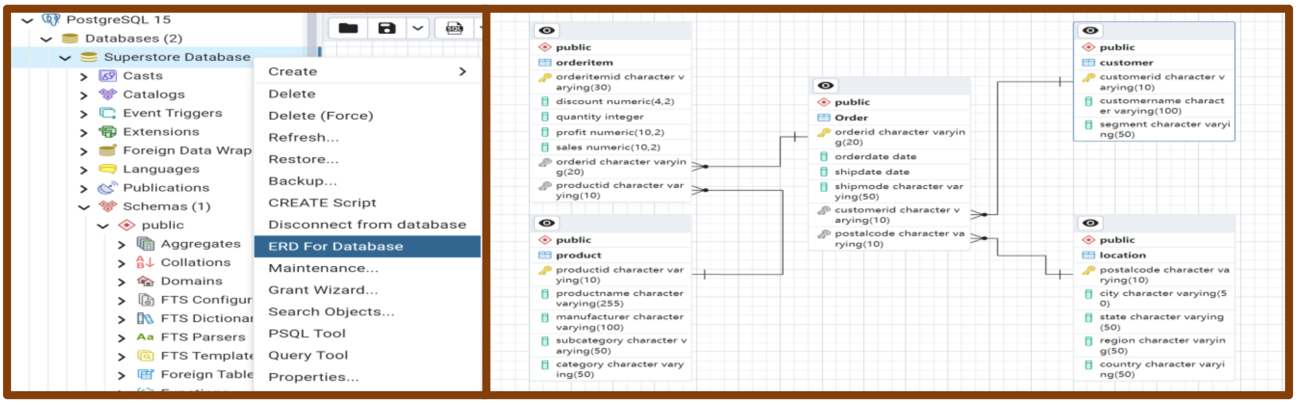
- Product.ProductID (PK) --> Order_item.ProductID (FK)

- **Order to Order_item Relationship:**
  - An Order may have one or more Orders_items; Each Order_item must belong to one and only one Order.
  - Order.OrderID (PK) --> Order_item.OrderID (FK)
- **Customer to Order Relationship:**
  - A Customer may have one or more Orders; Each Order must belong to one and only one Customer.
  - Customer.CustomerID (PK) --> Order.CustomerID (FK)
- **Location to Order Relationship:**
  - A PostalCode may belong to one or more Orders; Each Order must have one and only one PostalCode.
  - Location.PostalCode (PK) --> Order.PostalCode (FK)

3. Creating ERD

The ERD is constructed using PostgreSQL.

- **The 5 tables are interlinked using primary and foreign key relations.**



4. Creating Relational DB

Five tables have been established using PostgreSQL.

- **Order Table:**

Query | Query History

```
SELECT * FROM public."order"
ORDER BY orderid ASC LIMIT 100
```

Data Output | Messages | Notifications

	orderid [PK] characte	orderdate date	shipdate date	shipmode character var	customerid character var	postalcode character var
1	CA-2011-...	2011-09-07	2011-09-13	Standard ...	C00052	10024
2	CA-2011-...	2011-07-08	2011-07-12	Standard ...	C00392	94122
3	CA-2011-...	2011-03-14	2011-03-18	Standard ...	C00744	32216
4	CA-2011-...	2011-01-29	2011-02-04	Standard ...	C00749	10024
5	CA-2011-...	2011-04-08	2011-04-15	Standard ...	C00387	85301

- **Customer Table:**

Query Query History

```sql
SELECT * FROM public.customer
ORDER BY customerid ASC LIMIT 100
```

Data Output Messages Notifications

	customerid [PK] character varying (10)	customername character varying (100)	segment character varying (50)
1	C00001	William Brown	Consumer
2	C00002	Paul Prost	Home Office
3	C00003	Matt Abelman	Home Office
4	C00004	John Lee	Consumer
5	C00005	Seth Vernon	Consumer

- **Product Table:**

Query Query History

```sql
SELECT * FROM public.product
ORDER BY productid ASC LIMIT 100
```

Data Output Messages Notifications

	productid [PK] characte	productname character varying	manufacturer character vary	subcategory character var	category character var
1	L00001	While you Wer...	Other	Paper	Office Su...
2	L00002	#10- 4 1/8 x 9 ...	Other	Envelopes	Office Su...
3	L00003	#10- 4 1/8 x 9 ...	Other	Envelopes	Office Su...
4	L00004	#10 Gummed ...	Other	Envelopes	Office Su...
5	L00005	#10 Self-Seal ...	Other	Envelopes	Office Su...

- **Location Table:**

Query Query History

```sql
SELECT * FROM public.location
ORDER BY postalcode ASC LIMIT 100
```

Data Output Messages Notifications

	postalcode [PK] character varying	city character varying	state character varying	region character varying	country character varyi
1	10009	New York City	New York	East	United States
2	10011	New York City	New York	East	United States
3	10024	New York City	New York	East	United States
4	10035	New York City	New York	East	United States
5	1040	Holyoke	Massachusetts	East	United States

- **Order_item Table:**

Query Query History

```sql
SELECT * FROM public.orderitem
ORDER BY orderitemid ASC LIMIT 100
```

Data Output Messages Notifications

	orderitemid [PK] characte	discount numeric (4,2)	quantity integer	profit numeric (10,2	sales numeric (10,2	orderid character var	productid character
1	CA-2011-...	0.00	3	110.00	378.00	CA-2011-...	L00160
2	CA-2011-...	0.20	3	-88.00	502.00	CA-2011-...	L00817
3	CA-2011-...	0.20	6	69.00	197.00	CA-2011-...	L01639
4	CA-2011-...	0.20	6	32.00	91.00	CA-2011-...	L01667
5	CA-2011-...	0.20	1	1.00	4.00	CA-2011-...	L01311

5. 3NF Normalization

Customer, Location, Order, and Order_item relations are in 3NF

There is no partial functional and no transitive functional dependencies.

- **Customer (CustomerID, CustomerName, Segment)**
  - **FD1: CustomerID → CustomerName, Segment**
- **Location (PostalCode, City, State, Region, Country**
  - **FD1: PostalCode → City, State, Region, Country**
- **Order (OrderID, OrderDate, ShipDate, ShipMode, CustomerID(FK), PostalCode(FK))**
  - **FD1: OrderID → OrderDate, ShipDate, ShipMode, CustomerID(FK), PostalCode(FK)**
- **OrderItem (OrderItemID, Discount, Quantity, Profit, Sales, OrderID(FK), ProductID(FK))**
  - **FD1: OrderItemID → Discount, Quantity, Profit, Sales, OrderID(FK), ProductID(FK)**

Product is not in 3NF

Product is in 2NF since there is no partial functional dependencies, but there is transitive functional dependencies.

- **Product (ProductID, ProductName, Manufacturer, SubCategory, Category**
  - **FD1: ProductID → ProductName, Manufacturer, SubCategory, Category**
  - **FD2: SubCategory → Category**

Create a new relation for Category

- **Category (SubCategory, Category)**
  - **FD1: SubCategory → Category**
- **Product (ProductID, ProductName, Manufacturer, SubCategory)**
  - **FD1: ProductID → ProductName, Manufacturer, SubCategory**

6. Further Implementation

Final Relational Model is in 3NF

- **Customer (CustomerID, CustomerName, Segment**
  - **FD1: CustomerID → CustomerName, Segment)**
- **Location (PostalCode, City, State, Region, Country**
  - **FD1: PostalCode → City, State, Region, Country**
- **Order (OrderID, OrderDate, ShipDate, ShipMode, CustomerID(FK), PostalCode(FK)**
  - **FD1: OrderID → OrderDate, ShipDate, ShipMode, CustomerID(FK), PostalCode(FK)**
- **OrderItem (OrderItemID, Discount, Quantity, Profit, Sales, OrderID(FK), ProductID(FK)**
  - **FD1: OrderItemID → Discount, Quantity, Profit, Sales, OrderID(FK), ProductID(FK)**
- **Category (SubCategory, Category)**
  - **FD1: SubCategory → Category**
- **Product (ProductID, ProductName, Manufacturer, SubCategory)**
  - **FD1: ProductID → ProductName, Manufacturer, SubCategory**

Run a complex query in PostgreSQL

- **Calculate the average sales per order in Chicago and 2013 by customer segment.**

Query   Query History

```sql
SELECT
    Segment,
    AVG(sum_sales) AS AvgSalesPerOrder
FROM (
    SELECT
        o.OrderID,
        c.Segment,
        SUM(oi.Sales) AS sum_sales
    FROM
        "order" o
    JOIN
        OrderItem oi ON o.OrderID = oi.OrderID
    JOIN
        Customer c ON o.CustomerID = c.CustomerID
    JOIN
        Location l ON o.PostalCode = l.PostalCode
    WHERE
        l.City = 'Chicago'
        AND EXTRACT(YEAR FROM o.OrderDate) = 2013
    GROUP BY
        o.OrderID, c.Segment
) subquery
GROUP BY
    Segment;
```

Data Output   Messages   Graph Visualiser ×   Notifications

	segment character varying (50)	avgsalesperorder numeric
1	Consumer	411.5714285714285714
2	Corporate	76.0666666666666667
3	Home Office	547.3333333333333333

Compared to traditional Excel data storage, SQL offers several advantages:

- **SQL's structure is more organized than Excel's, making it easier for users to understand.**
- **SQL querying is more adaptable than Excel, accommodating a diverse range of data retrieval needs.**
- **SQL has fewer data redundancies, leading to storage efficiency and reduced inconsistencies.**
- **For processing large volumes of data, SQL databases outperform Excel, delivering faster data retrieval times.**
- **The data integrity and protection features in SQL are more robust, ensuring superior security against potential threats.**