**Group Project Technical Report**

Derrick Williams, Brandon Buser, Jason Kenosky, Magnas Saint-Hilaire, & Tracy Ficor

DBST652

Submitted To:

Dr. Marion Smith

Graduate School

University of Maryland University College

Date: 8/11/2019

**Table of Contents**

Contents

[**Group Project Technical Report** 1](#_heading=h.gjdgxs)

[Statement of Work or Introduction 3](#_heading=h.1fob9te)

[Software and Hardware 4](#_heading=h.1vsw3ci)

[Database Design 5](#_heading=h.3dy6vkm)

[Software Tools 6](#_heading=h.32hioqz)

[Entity Relationship Diagrams 7](#_heading=h.3q5sasy)

[Logical Model Diagram 7](#_heading=h.25b2l0r)

[Entities, Attributes and Relationships 7](#_heading=h.4fsjm0b)

[Relational Model Diagram 7](#_heading=h.2uxtw84)

[Tables, Attributes, and Relationships 7](#_heading=h.1a346fx)

[Data Definition Language (DDL) 8](#_heading=h.2iq8gzs)

[/\* Drop Objects \*/ 8](#_heading=h.xvir7l)

[/\* Create Tables \*/ 8](#_heading=h.3hv69ve)

[/\* Data Dictionary Query \*/ 8](#_heading=h.1x0gk37)

[/\* Create Constraints \*/ 8](#_heading=h.3u2rp3q)

[/\* Create Indexes \*/ 8](#_heading=h.2981zbj)

[Data Manipulation Language (DML) 9](#_heading=h.4anzqyu)

[Queries 9](#_heading=h.2pta16n)

[Performance and Maintenance Design Decisions 10](#_heading=h.3oy7u29)

[Security and Privacy Design Decisions 11](#_heading=h.2hio093)

[Database Management System Decisions 12](#_heading=h.odc9jc)

# Statement of Work - Introduction

# 

**Background:**

A large supply chain distributor, SUPERSTORE, wants to analyze their business operations by studying customer purchases and their demographics. The distributor recognizes that their existing operational database is not refined enough to undergo analysis, so they have decided that a new solution is needed. The solution is to design a brand new database that will be able to support studies and further analysis going forward.

**Objective:**

To implement a functional database that will provide the supply chain distributor the ability to analyze their operations in a new light that their existing operational database can’t provide.

**Scope:**

* Extract and analyze two data sets, Orders and Returns, from the existing operational database.
* Propose a database design that will successfully support the distributor’s end goal. The logical and relational designs will be created in Oracle SQL Developer Data Modeler.
* Upon finalization of the database design, the Data Definition Language (DDL) will be extracted from Oracle SQL Developer Data Modeler.
* The database will be created by executing the DDL in Oracle SQL Developer.
* The team will refine the Orders dataset to ensure the integrity of the data and then populate the clean data set in the new database.
* Constraints will be implemented to the database to ensure the integrity of the data and relieve some data maintenance issues for the distributor going forward.
* The team will run a variety of SQL commands against the database to test all constraints and to determine if the database is functional and will meet all requirements.

# 

# 

# 

# Software and Hardware

# 

Database:  
  
Oracle 12c, using Virtual Desktop Applications (VDA)  
  
Hardware and Software:  
  
UMUC (VDA) Intel(R) Xeon(R). Operating system is Windows 7.

The main software used in this project is Amazon Web Service, or AWS. AWS is a cloud computing technology which allows users with access to the Internet to connect to and utilize a virtual environment hosted by Amazon. Our individual connections to AWS were through our classroom LEO system In addition, AWS also allows for clients to store data on Amazon Servers freeing the need and costs associated with maintenance. Our own database that we had connected to is Oracle 12c, a free version of Oracle Database meant for use as a training database.

On top of AWS is a version of Windows 7 operating system which allows us to complete the project.

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Database Design

# Design of this database began with an Excel workbook containing the product and order information of SUPERSTORE. That data was converted to a single-table database for analysis of the existent data. Our goal was to identify any missing or redundant data and analyze the data set to determine an appropriate model for the data elements.

# 

# Various configurations were considered with different numbers of tables and distributions of the attributes of the original SUPERSTORE table. We decided against pursuing third-form normalization as a number of columns contained a limited subset of data and would likely prove inefficient as a larger number of table joins would be required in querying the data. Given the limitations of the original dataset that we were provided, a number of assumptions had to be made about the data going forward. Some of these assumptions will be discussed as the database design is described.

# 

# The first new table that we created was CUSTOMERS. This stores an ID number, first and last name, and segment for each customer. Segments are either Home Office, Corporate, or Consumer, and are included in this table as it is assumed that each customer is associated directly with a particular segment.

# 

# The second of our tables is ADDRESS. This table does not store full addresses, but rather includes just the postal code, city, state, region, and country of any distinct combination of postal code and city. That combination acts as the primary key of the table. Further specifics such as street addresses are not considered here, as they are tied to specific orders.

# 

# ORDERS is our third table. It includes an order ID number, order date, shipping date, shipping method, customer ID, street address, city, and postal code. Multiple products can be ordered together, but orders are assumed to have a single order date. Likewise, all products are assumed to be shipped together on a single data, via a single shipping method. Our data analysis found that all existing orders followed this pattern. We also assumed that customers can place separate orders to be shipped to separate addresses, so the street address of an order is included here, rather than being tied to the customer table.

# 

# The LINE\_ITEMS table includes an order ID, product ID, product name, quantity, and line item sale price. A single order can include multiple products within the order, so the combination of order ID, product ID, and product name for the primary key for this table. Analysis of the original data indicated that the cost of an individual product is determined by line item sale price divided by quantity, so future orders would need to follow that standard.

# 

# The final table derived from the SUPERSTORE data was PRODUCTS. The product IDs and product names are referenced in the line items table to determine when each product was ordered. This table consists of the product ID, category, sub-category, and product name. The existing data did include several examples of a product ID having two distinct product names associated with the same ID, so the combination of ID and name serve as the primary key here. Each product is assumed to remain within the same category and sub-category.

# 

# Additional data was provided about orders that were returned. Should that data need to be incorporated along with the original SUPERSTORE data, a RETURNS table would be utilized. Assumptions based on the data given indicate the only full orders can be returned, not subsets of products within the order, and that there are six specific categories tracked for why orders are returned. To allow a more customer-friendly return policy, information on product ID and name, quantity and item price, and return shipping date and method would need to be added.

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Software Tools

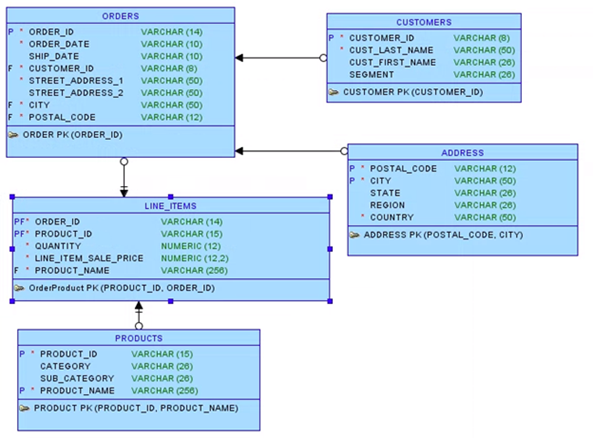
For software tools we used Oracle SQL Developer for SQL and DDL; Oracle Data Modeler to develop logical and relational models and also to produce DDM of the schema. For communication and collaboration we used Google Docs and Google Hangouts.

Individual hardware varied from Mac running a beta version of macOS to Windows -based desktops and laptops.

# Entity Relationship Diagrams

## 

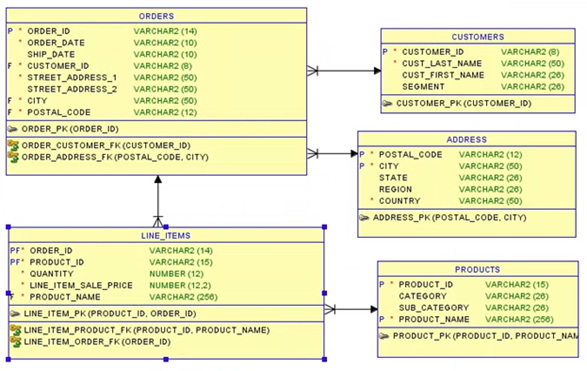
## Logical Model Diagram- Entities, Attributes and Relationships

****

## 

## 

## Relational Model Diagram - Tables, Attributes, and Relationships



# Data Definition Language (DDL)

## /\* Drop Objects \*/

DROP TABLE ADDRESS CASCADE CONSTRAINTS;

DROP TABLE CUSTOMERS CASCADE CONSTRAINTS;

DROP TABLE LINE\_ITEMS CASCADE CONSTRAINTS;

DROP TABLE ORDERS CASCADE CONSTRAINTS;

DROP TABLE PRODUCTS CASCADE CONSTRAINTS;

DROP TABLE RETURNS CASCADE CONSTRAINTS;

## /\* Create Tables \*/

/\* Table ADDRESS\*/

CREATE TABLE "ADDRESS"

( "POSTAL\_CODE" VARCHAR2(12 BYTE),

"CITY" VARCHAR2(50 BYTE),

"STATE" VARCHAR2(26 BYTE),

"REGION" VARCHAR2(26 BYTE),

"COUNTRY" VARCHAR2(50 BYTE)

) SEGMENT CREATION IMMEDIATE

PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255

NOCOMPRESS LOGGING

STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USERS" ;

/\*Table CUSTOMERS\*/

CREATE TABLE "CUSTOMERS"

( "CUSTOMER\_ID" VARCHAR2(8 BYTE),

"CUST\_LAST\_NAME" VARCHAR2(50 BYTE),

"CUST\_FIRST\_NAME" VARCHAR2(26 BYTE),

"SEGMENT" VARCHAR2(26 BYTE)

) SEGMENT CREATION IMMEDIATE

PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255

NOCOMPRESS LOGGING

STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USERS" ;

/\*Table LINE\_ITEMS\*/

CREATE TABLE "LINE\_ITEMS"

( "ORDER\_ID" VARCHAR2(14 BYTE),

"PRODUCT\_ID" VARCHAR2(15 BYTE),

"QUANTITY" NUMBER(12,0),

"LINE\_ITEM\_SALE\_PRICE" NUMBER(12,2),

"PRODUCT\_NAME" VARCHAR2(256 BYTE)

) SEGMENT CREATION IMMEDIATE

PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255

NOCOMPRESS LOGGING

STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USERS" ;

/\*Table ORDERS\*/

CREATE TABLE "ORDERS"

( "ORDER\_ID" VARCHAR2(14 BYTE),

"ORDER\_DATE" VARCHAR2(10 BYTE),

"SHIP\_DATE" VARCHAR2(10 BYTE),

"SHIP\_MODE" VARCHAR2(26 BYTE),

"CUSTOMER\_ID" VARCHAR2(8 BYTE),

"STREET\_ADDRESS\_1" VARCHAR2(50 BYTE),

"STREET\_ADDRESS\_2" VARCHAR2(50 BYTE),

"CITY" VARCHAR2(50 BYTE),

"POSTAL\_CODE" VARCHAR2(12 BYTE)

) SEGMENT CREATION IMMEDIATE

PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255

NOCOMPRESS LOGGING

STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USERS" ;

/\*Table PRODUCTS\*/

CREATE TABLE "PRODUCTS"

( "PRODUCT\_ID" VARCHAR2(15 BYTE),

"CATEGORY" VARCHAR2(26 BYTE),

"SUB\_CATEGORY" VARCHAR2(26 BYTE),

"PRODUCT\_NAME" VARCHAR2(256 BYTE)

) SEGMENT CREATION IMMEDIATE

PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255

NOCOMPRESS LOGGING

STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USERS" ;

/\*Table RETURNS\*/

CREATE TABLE "RETURNS"

( "ORDER\_ID" VARCHAR2(20 BYTE),

"REASON" VARCHAR2(64 BYTE)

) SEGMENT CREATION IMMEDIATE

PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255

NOCOMPRESS LOGGING

STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "USERS" ;

## /\* Data Dictionary Query \*/

SELECT OBJECT\_NAME, OBJECT\_TYPE FROM USER\_OBJECTS;

**/\* Create Constraints \*/**

# /\* CONSTRAINTS \*/

# /\* Constraints for Table ADDRESS \*/

# ALTER TABLE "ADDRESS" MODIFY ("POSTAL\_CODE" NOT NULL ENABLE);

# ALTER TABLE "ADDRESS" MODIFY ("CITY" NOT NULL ENABLE);

# ALTER TABLE "ADDRESS" MODIFY ("COUNTRY" NOT NULL ENABLE);

# ALTER TABLE "ADDRESS" ADD CONSTRAINT "ADDRESS\_PK" PRIMARY KEY ("POSTAL\_CODE", "CITY")

# USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

# STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

# PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

# BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

# TABLESPACE "USERS" ENABLE;

# ALTER TABLE "ADDRESS" ADD CONSTRAINT "CK\_REGION" CHECK (REGION IN ('East', 'West', 'Central', 'South')) ENABLE;

# /\* Constraints for Table CUSTOMERS \*/

# ALTER TABLE "CUSTOMERS" MODIFY ("CUSTOMER\_ID" NOT NULL ENABLE);

# ALTER TABLE "CUSTOMERS" MODIFY ("CUST\_LAST\_NAME" NOT NULL ENABLE);

# ALTER TABLE "CUSTOMERS" ADD CONSTRAINT "CUSTOMER\_PK" PRIMARY KEY ("CUSTOMER\_ID")

# USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

# STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

# PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

# BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

# TABLESPACE "USERS" ENABLE;

# ALTER TABLE "CUSTOMERS" ADD CONSTRAINT "CK\_SEGMENT" CHECK (SEGMENT IN ('Consumer','Corporate','Home Office')) ENABLE;

# /\* Constraints for Table LINE\_ITEMS \*/

# ALTER TABLE "LINE\_ITEMS" MODIFY ("ORDER\_ID" NOT NULL ENABLE);

# ALTER TABLE "LINE\_ITEMS" MODIFY ("PRODUCT\_ID" NOT NULL ENABLE);

# ALTER TABLE "LINE\_ITEMS" MODIFY ("QUANTITY" NOT NULL ENABLE);

# ALTER TABLE "LINE\_ITEMS" MODIFY ("LINE\_ITEM\_SALE\_PRICE" NOT NULL ENABLE);

# ALTER TABLE "LINE\_ITEMS" MODIFY ("PRODUCT\_NAME" NOT NULL ENABLE);

# ALTER TABLE "LINE\_ITEMS" ADD CONSTRAINT "LINE\_ITEM\_PK" PRIMARY KEY ("PRODUCT\_ID", "ORDER\_ID", “PRODUCT\_NAME”)

# USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

# STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

# PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

# BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

# TABLESPACE "USERS" ENABLE;

# ALTER TABLE "LINE\_ITEMS" ADD CONSTRAINT "CK\_QUANTITY" CHECK (QUANTITY > 0) ENABLE;

# ALTER TABLE "LINE\_ITEMS" ADD CONSTRAINT "CK\_PRICE" CHECK (LINE\_ITEM\_SALE\_PRICE > 0) ENABLE;

# /\* Constraints for Table ORDERS \*/

# ALTER TABLE "ORDERS" MODIFY ("ORDER\_ID" NOT NULL ENABLE);

# ALTER TABLE "ORDERS" MODIFY ("ORDER\_DATE" NOT NULL ENABLE);

# ALTER TABLE "ORDERS" MODIFY ("CUSTOMER\_ID" NOT NULL ENABLE);

# ALTER TABLE "ORDERS" MODIFY ("STREET\_ADDRESS\_1" NOT NULL ENABLE);

# ALTER TABLE "ORDERS" MODIFY ("CITY" NOT NULL ENABLE);

# ALTER TABLE "ORDERS" MODIFY ("POSTAL\_CODE" NOT NULL ENABLE);

# ALTER TABLE "ORDERS" ADD CONSTRAINT "ORDER\_PK" PRIMARY KEY ("ORDER\_ID")

# USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

# STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

# PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

# BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

# TABLESPACE "USERS" ENABLE;

# /\*Constraints for Table PRODUCTS\*/

# ALTER TABLE "PRODUCTS" MODIFY ("PRODUCT\_ID" NOT NULL ENABLE);

# ALTER TABLE "PRODUCTS" MODIFY ("PRODUCT\_NAME" NOT NULL ENABLE);

# ALTER TABLE "PRODUCTS" ADD CONSTRAINT "PRODUCT\_PK" PRIMARY KEY ("PRODUCT\_ID", "PRODUCT\_NAME")

# USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

# STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

# PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

# BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

# TABLESPACE "USERS" ENABLE;

# /\* REF CONSTRAINTS \*/

# /\* Ref Constraints for Table LINE\_ITEMS\*/

# ALTER TABLE "LINE\_ITEMS" ADD CONSTRAINT "LINE\_ITEM\_ORDER\_FK" FOREIGN KEY ("ORDER\_ID")

# REFERENCES "ORDERS" ("ORDER\_ID") ENABLE;

# ALTER TABLE "LINE\_ITEMS" ADD CONSTRAINT "LINE\_ITEM\_PRODUCT\_FK" FOREIGN KEY ("PRODUCT\_ID", "PRODUCT\_NAME")

# REFERENCES "PRODUCTS" ("PRODUCT\_ID", "PRODUCT\_NAME") ENABLE;

# /\*Ref Constraints for Table ORDERS\*/

# ALTER TABLE "ORDERS" ADD CONSTRAINT "ORDER\_ADDRESS\_FK" FOREIGN KEY ("POSTAL\_CODE", "CITY")

# REFERENCES "ADDRESS" ("POSTAL\_CODE", "CITY") ENABLE;

# ALTER TABLE "ORDERS" ADD CONSTRAINT "ORDER\_CUSTOMER\_FK" FOREIGN KEY ("CUSTOMER\_ID")

# REFERENCES "CUSTOMERS" ("CUSTOMER\_ID") ENABLE;

## /\* Create Indexes \*/

# /\*Index ADDRESS\_PK\*/

# 

# CREATE UNIQUE INDEX "ADDRESS\_PK" ON "ADDRESS" ("POSTAL\_CODE", "CITY")

# PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

# STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

# PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

# BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

# TABLESPACE "USERS" ;

# 

# /\* Index CUSTOMER\_PK\*/

# CREATE UNIQUE INDEX "CUSTOMER\_PK" ON "CUSTOMERS" ("CUSTOMER\_ID")

# PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

# STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

# PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

# BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

# TABLESPACE "USERS" ;

# 

# /\*Index LINE\_ITEM\_PK\*/

# CREATE UNIQUE INDEX "LINE\_ITEM\_PK" ON "LINE\_ITEMS" ("PRODUCT\_ID", "ORDER\_ID")

# PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

# STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

# PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

# BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

# TABLESPACE "USERS" ;

# /\*Index ORDER\_PK\*/

# 

# CREATE UNIQUE INDEX "ORDER\_PK" ON "ORDERS" ("ORDER\_ID")

# PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

# STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

# PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

# BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

# TABLESPACE "USERS" ;

# /\*Index PRODUCT\_PK\*/

# 

# CREATE UNIQUE INDEX "PRODUCT\_PK" ON "PRODUCTS" ("PRODUCT\_ID", "PRODUCT\_NAME")

# PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

# STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

# PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

# BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

# TABLESPACE "USERS" ;

# 

# Data Manipulation Language (DML)

## 

## /\* Queries \*/

**/\* TEST QUERIES\*/**

**-- Unique Key Test**

INSERT INTO PRODUCTS(PRODUCT\_ID, CATEGORY) VALUES (200001, 'Furniture');

**--Primary Key Test**

INSERT INTO ADDRESS (POSTAL\_CODE, CITY STATE ,REGION, COUNTRY) VALUES (57401, 'Aberdeen', 'Test', 'Test', 'Test');

INSERT INTO CUSTOMERS (CUST\_LAST\_NAME, CUST\_FIRST\_NAME, SEGMENT) VALUES (‘WILLIAMS’, ‘GEORGE’, ‘Customer’);

INSERT INTO LINE\_ITEMS (QUANTITY, LINE\_ITEM\_SALE\_PRICE, PRODUCT\_NAME) VALUES (2, 245.00, ‘BIC Ultra Round Stick Grip’);

INSERT INTO ORDERS (ORDER\_DATE, SHIP\_DATE, SHIP\_MODE) VALUES (‘8-5-2019’, ‘8-6-2019’, ‘First Class’, );

**-- Foreign Key Test**

INSERT INTO LINE\_ITEMS (ORDER\_ID, PRODUCT\_ID, QUANTITY, LINE\_ITEM\_SALE\_PRICE)

VALUES ('CA-2011-114195','OFF-BI-11111494', 'Test', 5);

INSERT INTO ORDERS (ORDER\_ID, ORDER\_DATE, SHIP\_DATE, SHIP\_MODE)

VALUES ('CA-2011-114195','8-5-2019’, ‘8-6-2019’, ‘First Class’);

**--Check Constraint**

INSERT INTO LINE\_ITEMS (ORDER\_ID, PRODUCT\_ID, PRODUCT\_NAME, QUANTITY, LINE\_ITEM\_SALE\_PRICE) VALUES ('CA-2012-136469', 'OFF-BI-10000494', 'Acco Economy Flexible Poly Round Ring Binder', -5, 5);

**/\* Not Null constraint on CUSTOMERS, ADDRESS, ORDERS, LINE\_ITEM, AND PRODUCTS tables\*/**

INSERT INTO CUSTOMERS (CUSTOMER\_ID) VALUES ('AA-TT111');

INSERT INTO ADDRESS (POSTAL\_CODE) VALUES ('85201');

INSERT INTO ORDERS(ORDER\_DATE) VALUES (‘3-5-2018');

INSERT INTO PRODUCT(PRODUCT\_NAME) VALUES ('BIC Ultra Round Stick Grip');

INSERT INTO LINE\_ITEM(PROUCT\_ID) VALUES ('542013’)

**/\* DATABASE QUERIES PART 3 \*/**

SET ECHO ON;

**--What are the names of the customers who have returned products?**

SELECT DISTINCT CUST\_FIRST\_NAME, CUST\_LAST\_NAME

FROM CUSTOMERS C JOIN ORDERS O

ON C.CUSTOMER\_ID = O.CUSTOMER\_ID

WHERE O.ORDER\_ID IN

(SELECT ORDER\_ID FROM RETURNS);

**--What customers have not ordered any products?**

SELECT CUSTOMER\_ID, CUST\_FIRST\_NAME, CUST\_LAST\_NAME

FROM CUSTOMERS

WHERE CUSTOMER\_ID NOT IN

(SELECT CUSTOMER\_ID FROM ORDERS);

**--What are the most profitable products ordered?**

SELECT \* FROM (SELECT PRODUCT\_ID, PRODUCT\_NAME,

SUM(LINE\_ITEM\_SALE\_PRICE) AS TOT\_SALES

FROM LINE\_ITEMS

GROUP BY PRODUCT\_ID, PRODUCT\_NAME

ORDER BY TOT\_SALES DESC)

WHERE ROWNUM < 11;

**--Which customers are repeat customers?**

SELECT CUSTOMER\_ID, CUST\_FIRST\_NAME, CUST\_LAST\_NAME FROM

CUSTOMERS WHERE CUSTOMER\_ID IN (SELECT CUSTOMER\_ID FROM

(SELECT CUSTOMER\_ID, COUNT(\*) FROM ORDERS GROUP BY CUSTOMER\_ID

HAVING COUNT(\*) > 1));

**--What order is the second most profitable?**

WITH secondMost

AS ( SELECT ORDER\_ID, Total, ROW\_NUMBER() OVER(ORDER BY Total DESC) AS Row\_Number FROM ( SELECT distinct p.ORDER\_ID, SUM(LINE\_ITEM\_SALE\_PRICE) OVER(PARTITION BY p.ORDER\_ID) AS Total FROM Sales sa INNER JOIN ORDERS p ON sa.ORDER\_ID = p.ORDER\_ID ORDER BY Total desc) ORDER BY Row\_Number )

SELECT \* FROM secondMost

WHERE Row\_Number = 2;

**--How many orders has each customer placed?**

SELECT CUSTOMER\_ID, COUNT(ORDER\_ID) TOT\_ORDERS

FROM ORDERS GROUP BY CUSTOMER\_ID;

**--On average, how many line items are on an order?**

SELECT COUNT(ORDER\_ID) / COUNT(DISTINCT ORDER\_ID) AVG\_LINE\_ITEMS

FROM LINE\_ITEMS;

**--What product is most frequently ordered?**

SELECT Q1.\* FROM (

SELECT PRODUCT\_ID, PRODUCT\_NAME, COUNT(\*) PROD\_CT

FROM LINE\_ITEMS

GROUP BY PRODUCT\_ID, PRODUCT\_NAME) Q1,

(SELECT MAX(Q2.PROD\_CT) HIGH\_CT FROM (

SELECT PRODUCT\_ID, PRODUCT\_NAME, COUNT(\*) PROD\_CT

FROM LINE\_ITEMS

GROUP BY PRODUCT\_ID, PRODUCT\_NAME) Q2) Q3

WHERE Q1.PROD\_CT = Q3.HIGH\_CT;

**/\*Database Queries PART 4\*/**

**--Query 1: What is the total sales amount for all products under the Furniture category?**

SELECT CATEGORY, ROUND(SUM(LINE\_ITEM\_SALE\_PRICE),2) TOTAL\_SALES

FROM LINE\_ITEMS LI JOIN PRODUCTS P

ON LI.PRODUCT\_ID = P.PRODUCT\_ID

AND LI.PRODUCT\_NAME = P.PRODUCT\_NAME

WHERE CATEGORY = 'Furniture'

GROUP BY CATEGORY;

**--Query 2: What is the total sales amount for each category?**

SELECT category, ROUND(SUM(line\_item\_sale\_price ),2) total\_sales

FROM line\_items

INNER JOIN products

USING(product\_id, product\_name)

GROUP BY category

ORDER BY total\_sales;

**--Query 3: What is the total sales amount for each category and grand total sales amount for all categories?**

SELECT CATEGORY, ROUND(SUM(LINE\_ITEM\_SALE\_PRICE),2) TOTAL\_SALES

FROM LINE\_ITEMS LI JOIN PRODUCTS P

ON LI.PRODUCT\_ID = P.PRODUCT\_ID

AND LI.PRODUCT\_NAME = P.PRODUCT\_NAME

GROUP BY ROLLUP (CATEGORY);

**--Query 4: What is the total sales amount for each subcategory and grand total sales amount for all subcategories and categories?**

SELECT CATEGORY, SUB\_CATEGORY, ROUND(SUM(LINE\_ITEM\_SALE\_PRICE),2) TOTAL\_SALES

FROM LINE\_ITEMS LI JOIN PRODUCTS P

ON LI.PRODUCT\_ID = P.PRODUCT\_ID

AND LI.PRODUCT\_NAME = P.PRODUCT\_NAME

GROUP BY ROLLUP (CATEGORY, SUB\_CATEGORY);

**--Query 5: What is the customer's total sales amount for each subcategory and grand total sales amount for all categories?**

SELECT CUSTOMER\_ID, CATEGORY, SUB\_CATEGORY, ROUND(SUM(LINE\_ITEM\_SALE\_PRICE),2) TOTAL\_SALES

FROM LINE\_ITEMS LI JOIN PRODUCTS P

ON LI.PRODUCT\_ID = P.PRODUCT\_ID

AND LI.PRODUCT\_NAME = P.PRODUCT\_NAME

JOIN ORDERS O ON O.ORDER\_ID = LI.ORDER\_ID

GROUP BY ROLLUP (CUSTOMER\_ID, CATEGORY, SUB\_CATEGORY);

**--Query 6: How many orders are there by customer and what is the total sales sorted by customer's total number of orders?**

SELECT customer\_id, COUNT(order\_id) sum\_of\_orders,  
 ROUND(SUM(line\_item\_sale\_price ),2) total\_sales

FROM line\_items

JOIN orders

USING(order\_id)

GROUP BY customer\_id

ORDER BY sum\_of\_orders desc;

**--Query 7: How many orders are there by customer and what is the total sales sorted by customer's total sales amount?**

SELECT CUSTOMER\_ID, COUNT(O.ORDER\_ID) NUMBER\_OF\_ORDERS, ROUND(SUM(LINE\_ITEM\_SALE\_PRICE),2) TOTAL\_SALES

FROM LINE\_ITEMS LI JOIN ORDERS O

ON O.ORDER\_ID = LI.ORDER\_ID

GROUP BY CUSTOMER\_ID

ORDER BY TOTAL\_SALES DESC;

**--Query 8: What are the subtotals of customers for the combination of state and city?**

SELECT A.STATE CUSTOMER\_STATE, A.CITY CUSTOMER\_CITY, COUNT(DISTINCT CUSTOMER\_ID) NUMBER\_OF\_CUSTOMERS

FROM ADDRESS A JOIN ORDERS O

ON A.POSTAL\_CODE = O.POSTAL\_CODE AND A.CITY = O.CITY

GROUP BY CUBE (A.CITY, A.STATE)

ORDER BY CUSTOMER\_STATE DESC, CUSTOMER\_CITY DESC;

**--Query 9: Find the subtotals of customers for the combination of state and city and --replace null values by ========**

SELECT NVL(state, '=======') customer\_state, NVL(city, '=======') customer\_city, count(distinct customer\_id) number\_of\_customers

FROM address

JOIN orders

USING(postal\_code, city)

GROUP BY ROLLUP(city, state)

ORDER BY customer\_state, customer\_city;

**--Query 10: For each customer in the customer table, return the total number as a running total.**

WITH RUNNING\_TOTAL\_ORDERS

AS (

SELECT DISTINCT C.CUSTOMER\_ID, CONCAT(TRIM(C.CUST\_FIRST\_NAME), CONCAT(' ', C.CUST\_LAST\_NAME)) AS CUSTOMER\_NAME, O.ORDER\_ID, O.ORDER\_DATE

FROM CUSTOMERS C INNER JOIN ORDERS O

ON C.CUSTOMER\_ID = O.CUSTOMER\_ID

)

SELECT A.\*, COUNT(CUSTOMER\_NAME) OVER (PARTITION BY CUSTOMER\_NAME ORDER BY CUSTOMER\_NAME, ORDER\_ID, ORDER\_DATE) AS "RUNNING TOTAL OF ORDERS"

FROM RUNNING\_TOTAL\_ORDERS A

ORDER BY CUSTOMER\_NAME ASC;

**--Query 11: What is the count of customers in CUSTOMERS table that is not found in ORDERS table?**

SELECT COUNT(\*) FROM (

SELECT DISTINCT CUSTOMER\_ID FROM CUSTOMERS

MINUS

SELECT DISTINCT CUSTOMER\_ID FROM ORDERS);

**--Query 12: What is the count of customers in CUSTOMERS table that is found in ORDERS table?**

SELECT COUNT(\*) FROM (

SELECT DISTINCT CUSTOMER\_ID FROM CUSTOMERS

INTERSECT

SELECT DISTINCT CUSTOMER\_ID FROM ORDERS);

**--Query 13: What is the average product price for each product sub category?**

# SELECT products.category, products.sub\_category, ROUND(AVG(line\_item\_sale\_Price),2)

# FROM products

# INNER JOIN

# line\_items ON products.product\_id = line\_items.product\_id

# GROUP BY products.category, products.sub\_category;

**--Query 14: What is the total sales for each region and subtotal for each state?**

SELECT REGION, STATE, SUM(LINE\_ITEM\_SALE\_PRICE) TOTAL\_SALES

FROM ADDRESS A JOIN ORDERS O

ON A.POSTAL\_CODE = O.POSTAL\_CODE AND A.CITY = O.CITY

JOIN LINE\_ITEMS LI ON O.ORDER\_ID = LI.ORDER\_ID

GROUP BY REGION, ROLLUP (STATE);

**--Query 15: What is the total number of orders by shipping class?**

SELECT SHIP\_MODE, COUNT(DISTINCT ORDER\_ID)

FROM ORDERS

GROUP BY SHIP\_MODE;

**--Query 16: What are the monthly total sales from 2013 to 2014?**

SELECT ORDER\_MONTH, ORDER\_YEAR, SUM(LINE\_ITEM\_SALE\_PRICE) FROM

(SELECT SUBSTR(ORDER\_DATE,1,2) ORDER\_MONTH, SUBSTR(ORDER\_DATE,7,4) ORDER\_YEAR, LINE\_ITEM\_SALE\_PRICE

FROM LINE\_ITEMS LI JOIN ORDERS O

ON LI.ORDER\_ID = O.ORDER\_ID) S

WHERE ORDER\_YEAR IN (2013,2014)

GROUP BY ORDER\_MONTH, ORDER\_YEAR

ORDER BY ORDER\_YEAR, ORDER\_MONTH;

**--Query 17: What customers have same day shipping from 2013 to 2016?**

SELECT DISTINCT CUSTOMER\_ID FROM ORDERS

WHERE SHIP\_MODE = 'Same Day'

AND SUBSTR(ORDER\_DATE,7,4) IN (2013,2014,2015,2016);

**--Query 18: Which customers have ordered in 2014 but did not order in 2015?**

SELECT CUSTOMER\_ID

FROM ORDERS

WHERE SUBSTR(ORDER\_DATE, 7, 4) = 2014

MINUS

SELECT CUSTOMER\_ID FROM ORDERS

WHERE SUBSTR(ORDER\_DATE, 7, 4) = 2015;

# [Performance and Maintenance Design Decisions](#_heading=h.38czs75)

# 

Performance tuning requires a different, although related, method to the initial configuration of a system. Configuring a system involves allocating resources in an ordered manner so that the initial system configuration is functional. Tuning is driven by identifying the most significant bottleneck and making the appropriate changes to reduce or eliminate the effect of that bottleneck. Usually, tuning is performed reactively, either while the system is in pre-production or after it is in production.  
**Ideally, baseline data gathered should include the following:**

* Application statistics (transaction volumes, response time)
* Database statistics
* Operating system statistics
* Disk I/O statistics
* Network statistics

Additionally, to keep the system running optimally, proactive monitoring on a regularly scheduled interval will be required, several performance statistics will need to be examined to identify whether the system behavior and resource usage have changed. Proactive monitoring can also be considered as proactive tuning.  
  
 Oracle Database provides several tools to gather information regarding database performance. In addition to gathering data, Oracle Database provides the following tools that can be used to monitor performance, diagnose problems, and tuning of our newly created application.

* Automatic Workload Repository (AWR) - <https://docs.oracle.com/cd/E11882_01/server.112/e41573/autostat.htm#i27008>
* Automatic Database Diagnostic Monitor - (ADDM) <https://docs.oracle.com/cd/E11882_01/server.112/e41573/autostat.htm#i27008>
* SQL Tuning Advisor - <https://docs.oracle.com/cd/E11882_01/server.112/e41573/sql_tune.htm#i34782>
* Oracle views/ V$ Performance Views - <https://docs.oracle.com/cd/E11882_01/server.112/e41573/instance_tune.htm#g58652>

Furthermore, Oracle Database has provided us with many ways to automate several of the common maintenance tasks that we will need to perform. These automated maintenance tasks can be performed when the system load is expected to be light. We can enable and disable individual maintenance tasks and can configure when these tasks run and what resource allocations they are allotted.

**Some of these predefined automated maintenance tasks are:**

* Automatic Segment Advisor—Identifies segments that have space available for reclamation, and makes recommendations on how to defragment those segments.
* Automatic SQL Tuning Advisor—Examines the performance of high-load SQL statements, and makes recommendations on how to tune those statements. You can configure this advisor to automatically implement SQL profile recommendations.
* SQL Plan Management (SPM) Evolve Advisor—Evolves plans that have recently been added to the SQL plan baseline. The advisor simplifies plan evolution by eliminating the requirement to do it manually.
* Performing and Scheduling Backups with Enterprise Manager - Enterprise Manager lets you perform all of the different backup types supported by RMAN, and schedule the different backup jobs required by your backup strategy.

# [Security and Privacy Design Decisions](#_heading=h.1nia2ey)

Security breaches can attack any vulnerability found in an environment. To ensure the security and privacy of our users, we will need to  ensure to protect our networks with firewalls, use secure backups, take advantage of the many features offered by Oracle, such as:

* Oracle Database firewall - The Oracle database firewall option prevents database attacks from internal and external accesses by acting as the first line of database defense. It sits on the network layer, transporting data to the application and database layers without any application modification needed
* Data Masking - Data is obscured by random characters or other datum. The data can be masked either through an Oracle Enterprise Manager Cloud Control or with Data Pumps
* Advance Security (TDE, Data Redaction) - A method used to encrypt the data in the database, TDE will protect from users steal the physical data files. TDE is transparent and can be applied at column level or at the entire tablespace level.
* Label security - Enables companies and government organizations to consolidate data with different access requirements (including government classified data) into the same database.  It implements multilevel access controls based on the classification of the data and the access label of the application user.

In addition to those features, to prevent unauthorized use of a database several other features and security mechanism can be implemented, such as; Virtual Private Database (VPD), Fine-Grained Auditing (FGA), Data Vault, Database auditing, Audit Vault, Storage/Network authentication, Secure Socket Layer (SSL), Kerberos and CyberSafe with Oracle Advanced Security protocol, Entrust/PKI Support in Oracle Advanced Security and Password-Authenticated Enterprise Users.

# Database Management Systems Decisions

* The distributor will need to decide if they want to integrate their existing operational database with the new database where the new database serves as a data warehouse for analysis, or have the new database replace their operational database completely. If the distributor selects the latter, we recommend looking into a Business Intelligence platform to automate their analysis/reports.
* We recognize the data discrepancies from the existing operation database, and with this the team recommends to implement a change management process for the new database. The process will ensure that all changes to the database going forward will be vetted by admins and communicated with all users. This will prevent any lapse in maintaining the database and ensure that the data quality is high.
* The team also recommends that the database is hosted by a cloud based web server in the future. This will relieve the distributor’s technological staff of data center/hosting duties and they can focus on the actual functions of the database.