



KGSB Clothing

Knoelle Grassi & Samantha Bentley
kgrassi2693 & sbentley4059
kgrassi2693_kgsb_clothing_db_project

Revolutionizing KGSB Clothing: Solving Inventory and Data Management Challenges

KGSB Clothing seeks to build a database to manage their data. The *current challenge* lies in the absence of a robust database system to efficiently manage inventory, track customer information, and ensure data integrity. To *address this*, we propose implementing a database solution that streamlines inventory management, maintains detailed customer records, and ensures data integrity through a robust design. This solution will facilitate efficient data access, enabling quick retrieval of information for order processing, reporting, and analytics, ultimately enhancing the overall functionality and customer experience of the platform.

Optimizing Operations: Solving Inventory and Customer Management Challenges

Our company's **problems** include inventory management, managing customer information, data integrity, and efficient data access. We need an efficient way to track the inventory of our store. This includes maintaining accurate records of product quantities, restocking, and handling returns. We also need to manage customer data. We must keep track of customer profiles, purchase history, and other relevant information. We must ensure data consistency and prevent duplicate entries to ensure business operations run smoothly. The store's database should also allow quick and easy information retrieval for order processing, reporting, and analytics purposes.

Our **objectives** are to create a product catalog, track inventory, create a customer catalog, order processing, and create reports on our store. The product catalog should show details about every product. It requires the name, description, price, category, and availability of each product. The inventory tracking should maintain real-time inventory records. Whenever a product is sold or restocked, the inventory should be updated accordingly. The customer catalog should keep track of customer information. It should maintain personal details like name, contact information, and addresses of every customer along with purchase history with orders and products bought. Through order processing the database should keep track of incoming inventory and who the incoming inventory is from. The database should support various reports such as sales summaries, stock levels, and customer information.

Our **constraints** are budget, scalability, security, performance, and readability. The database needs to be built within budget constraints. It also should accommodate future growth without major overhauls. There should also be proper access controls and encryption to protect Customer data. The database should also be able to perform well even during peak times. The database should be designed where it's easy to read and maintain by anyone who requires access.

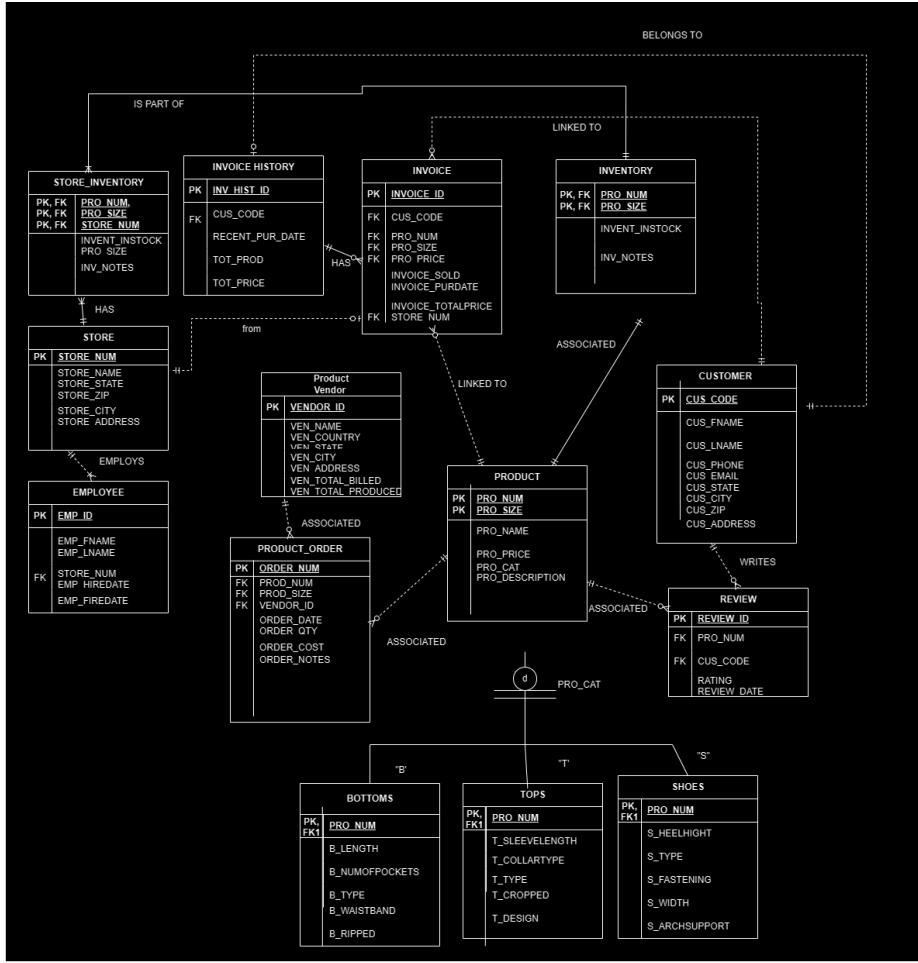
Scope, Boundaries, and Business Rules:

Our **Scope** involves Inventory Management, Customer Information Management, Data Integrity, and Efficient Data Access. Inventory management should track product quantities, restocking, and handling returns. Customer Information Management should maintain customer profiles, purchase history, and other relevant information for the customer. Data Integrity should ensure consistency and prevent duplicate entries. Efficient Data access should have quick and easy access to get information for order processing, reporting, and analytics.

We should have entities for Product, Inventory, Invoice, Customer, Product Vendor, Invoice History, Review, Employee, Stores, Store Inventory, and Product Order.

The **Product** entities should have subtypes of *bottoms*, *tops*, and *shoes*. It should have attributes of product number, product name, product price, product category, product size, and product description. The product category should be the subtype discriminator for the subtypes. *Bottoms* should have attributes of length, number of pockets, type, waistband, and if it's ripped. *Tops* should have attributes of sleeve length, collar type, type, if it's cropped, and design. *Shoes* should have attributes of heel height, shoe type, fastenings, width, and arch support. The **Inventory** entities should have attributes for product number and product size from product table, quantity in stock, and inventory notes. The **Invoice** entities should have attributes for customer code from customer, product number and product size from product, number of products sold, purchase date, product price from product, total price which is derived from qty sold and product price, invoice number, and store number. The **Invoice History** entities should have attributes customer code from Invoice, total products bought, total price, and invoice history id. The **Customer** entities should have attributes for customer first name, customer last name, customer phone number, customer email, customer state, customer city, customer zip code, customer street number, and customer code. The **product vendor** entities should have attributes for vendor number, vendor name, vendor country, vendor state, vendor city, vendor address, vendor total billed, and vendor total produced. The **Product Order** entities should have attributes for order number, product number and product size from product, vendor number from product vendor, order date, order quantity, order cost, and order notes. The **Review** entities should have attributes for review ID, product number and product size from product, customer code from customer, rating, and review date. The **Employee** entities should have attributes for employee number, employee first name, employee last name, employee hire date, employee fire date, and store number from the store they're employed at from store. The **Store** entities should have attributes for store number, store name, store state, store ZIP code, store state, store city, and store address. Store Inventory

Each Product can be categorized as Bottoms, Tops, or Shoes. Each Inventory record is associated with one Product. Each Invoice is linked to one Customer and one product. Each Invoice History record is linked to one Customer. Each product is associated with many product orders. Each product order is associated with one product vendor. Each Review is associated with one Product and written by one Customer. Each Employee works at one Store. Each Store can have multiple Store Inventory records, each representing a stocked Product. Each invoice is from a store.



```

1  -- MariaDB dump 10.19 Distrib 10.9.8-MariaDB, for Linux (x86_64)
2  --
3  -- Host: 10.200.208.126   Database: kgrassi2693_kgsb_clothing_db_project
4  --
5  -- Server version 10.4.33-MariaDB
6
7  /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
8  /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
9  /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
10 /*!40101 SET NAMES utf8mb4 */;
11 /*!40103 SET @OLD_TIME_ZONE=@TIME_ZONE */;
12 /*!40103 SET TIME_ZONE='+00:00' */;
13 /*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
14 /*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
15 /*!40101 SET @OLD_SQL_MODE=@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
16 /*!40111 SET @OLD_SQL_NOTES=@SQL_NOTES, SQL_NOTES=0 */;

17
18
19 -- Table structure for table `BOTTOMMS`
20 --
21
22 DROP TABLE IF EXISTS `BOTTOMMS`;
23 /*!40101 SET @saved_cs_client      = @@character_set_client */;
24 /*!40101 SET character_set_client = utf8 */;
25 CREATE TABLE `BOTTOMMS` (
26     `PROD_NUM` int(11) NOT NULL,
27     `PROD_SIZE` varchar(6) NOT NULL,
28     `LENGTH` varchar(50) DEFAULT NULL,
29     `NUM_OF_POCKETS` int(11) DEFAULT NULL,
30     `TYPE` varchar(50) NOT NULL,
31     `WAISTBAND` varchar(50) DEFAULT NULL,
32     `IS_RIPPED` tinyint(1) DEFAULT NULL,
33     KEY `PROD_NUM` (`PROD_NUM`,`PROD_SIZE`),
34     CONSTRAINT `BOTTOMMS_ibfk_1` FOREIGN KEY (`PROD_NUM`, `PROD_SIZE`) REFERENCES `PRODUCT` (`PROD_NUM`, `PROD_SIZE`)
35 ) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;
36 /*!40101 SET character_set_client = @saved_cs_client */;

37
38
39 -- Dumping data for table `BOTTOMMS`
40 --
41
42 LOCK TABLES `BOTTOMMS` WRITE;
43 /*!40000 ALTER TABLE `BOTTOMMS` DISABLE KEYS */;
44 INSERT INTO `BOTTOMMS` VALUES
45 (1,'32','Ankle',4,'Jeans','Normal',1),
46 (5,'30','Mid Thigh',3,'Shorts','High-Waisted',0),
47 (1,'38','Ankle',4,'Jeans','Normal',1),
48 (5,'40','Mid Thigh',3,'Shorts','High-Waisted',0),
49 (10,'28','Mini',3,'Skirt','High-Waisted',0),
50 (14,'M','Ankle',0,'Leggings','Normal',0),
51 (1,'30','Ankle',4,'Jeans','Normal',1),
52 (5,'44','Mid Thigh',3,'Shorts','High-Waisted',0),
53 (10,'34','Mini',3,'Skirt','High-Waisted',0),
54 (14,'L','Ankle',0,'Leggings','Normal',0),
55 (20,'36','Ankle',100,'Cargo Pants','High-Waisted',0),
56 /*!40000 ALTER TABLE `BOTTOMMS` ENABLE KEYS */;
57 UNLOCK TABLES;
58
59
60 -- Table structure for table `CUSTOMER`
61 --
62
63 DROP TABLE IF EXISTS `CUSTOMER`;
64 /*!40101 SET @saved_cs_client      = @@character_set_client */;
65 /*!40101 SET character_set_client = utf8 */;
66 CREATE TABLE `CUSTOMER` (
67     `CUS_CODE` int(11) NOT NULL AUTO_INCREMENT,
68     `CUS_FNAME` varchar(50) DEFAULT NULL,
69     `CUS_LNAME` varchar(50) NOT NULL,
70     `CUS_PHONE` bigint(10) DEFAULT NULL,

```

Invoice based on the customer with the first name Emerald and the purchase date 1991-10-31.

```
1345 ✓ SELECT INVOICE_ID, CUS_FNAME, CUS_LNAME, INVOICE.CUS_CODE AS CUS_CODE, PROD_NUM, PROD_SIZE, QTY SOLD,
1346     PURCHASE_DATE, TOTAL_PRICE, STORE_ID FROM INVOICE, CUSTOMER
1347 WHERE CUS_FNAME = 'Emerald' AND PURCHASE_DATE = '1991-10-31' AND CUSTOMER.CUS_CODE = INVOICE.CUS_CODE;
```

Result 686

EN	INVOICE_ID	CUS_FNAME	CUS_LNAME	CUS_CODE	PROD_NUM	PROD_SIZE	QTY SOLD	PURCHASE_DATE	TOTAL_PRICE	STORE_ID
1		1 Emerald	Kerluke		3	1.30		33 1991-10-31	329.67	3

Invoice based on the customer with the first name Virginia and the purchase date 2014-11-16.

```
1347 ✓ SELECT INVOICE_ID, CUS_FNAME, CUS_LNAME, INVOICE.CUS_CODE AS CUS_CODE, PROD_NUM, PROD_SIZE, QTY SOLD,
1348     PURCHASE_DATE, TOTAL_PRICE, STORE_ID FROM INVOICE, CUSTOMER
1349 WHERE CUS_FNAME = 'Virginia' AND PURCHASE_DATE = '2014-11-16' AND CUSTOMER.CUS_CODE = INVOICE.CUS_CODE;
```

Result 688

EN	INVOICE_ID	CUS_FNAME	CUS_LNAME	CUS_CODE	PROD_NUM	PROD_SIZE	QTY SOLD	PURCHASE_DATE	TOTAL_PRICE	STORE_ID
1		91 Virginia	Kilback		91	1.30		99 2014-11-16	989.01	18

Invoice based on the customer with the first name Ella and the purchase date 2019-01-04.

```
1349 ✓ SELECT INVOICE_ID, CUS_FNAME, CUS_LNAME, INVOICE.CUS_CODE AS CUS_CODE, PROD_NUM, PROD_SIZE, QTY SOLD,
1350     PURCHASE_DATE, TOTAL_PRICE, STORE_ID FROM INVOICE, CUSTOMER
1351 WHERE CUS_FNAME = 'Ella' AND PURCHASE_DATE = '2019-01-04' AND CUSTOMER.CUS_CODE = INVOICE.CUS_CODE;
```

Result 690

EN	INVOICE_ID	CUS_FNAME	CUS_LNAME	CUS_CODE	PROD_NUM	PROD_SIZE	QTY SOLD	PURCHASE_DATE	TOTAL_PRICE	STORE_ID
1		32 Ella	Collins		12	6.8.5		52 2019-01-04	1143.48	7

The current inventory of products sold within this year so far, 2024-01-01 to 2024-04-04.

```
1353 ✓ SELECT INVENTORY.PROD_NUM, INVENTORY.PROD_SIZE, INVENT_INSTOCK FROM INVENTORY, INVOICE  
1354 WHERE PURCHASE_DATE BETWEEN '2024-01-01' AND '2024-04-04'  
1355 AND INVENTORY.PROD_NUM = INVOICE.PROD_NUM AND INVENTORY.PROD_SIZE = INVOICE.PROD_SIZE  
1356 GROUP BY PROD_NUM, PROD_SIZE;
```

	PROD_NUM	PROD_SIZE	INVENT_INSTOCK
1	18	9.5	3281

The current inventory of products sold in a year, 2023-04-04 to 2024-04-04.

```
1358 ✓ SELECT INVENTORY.PROD_NUM, INVENTORY.PROD_SIZE, INVENT_INSTOCK FROM INVENTORY, INVOICE  
1359 WHERE PURCHASE_DATE BETWEEN '2023-04-04' AND '2024-04-04'  
1360 AND INVENTORY.PROD_NUM = INVOICE.PROD_NUM AND INVENTORY.PROD_SIZE = INVOICE.PROD_SIZE  
1361 GROUP BY PROD_NUM, PROD_SIZE;
```

	PROD_NUM	PROD_SIZE	INVENT_INSTOCK
1	6	8.5	799730
2	18	9.5	3281
3	20	36	9980

The current inventory of products sold in a calendar year, 2023-01-01 to 2023-12-31.

```
1363 ✓ SELECT INVENTORY.PROD_NUM, INVENTORY.PROD_SIZE, INVENT_INSTOCK FROM INVENTORY, INVOICE  
1364 WHERE PURCHASE_DATE BETWEEN '2023-01-01' AND '2023-12-31'  
1365 AND INVENTORY.PROD_NUM = INVOICE.PROD_NUM AND INVENTORY.PROD_SIZE = INVOICE.PROD_SIZE  
1366 GROUP BY PROD_NUM, PROD_SIZE;
```

	PROD_NUM	PROD_SIZE	INVENT_INSTOCK
1	6	8.5	799730
2	16	M	91
3	20	36	9980

Inventory of all products that are the type Shoes ('S').

```
1368 ✓ SELECT INVENTORY.PROD_NUM, INVENTORY.PROD_SIZE, INVENT_INSTOCK
1369     FROM INVENTORY, PRODUCT
1370     WHERE PROD_CAT = 'S' AND INVENTORY.PROD_NUM = PRODUCT.PROD_NUM AND INVENTORY.PROD_SIZE = PRODUCT.PROD_SIZE;
```

	PROD_NUM	PROD_SIZE	INVENT_INSTOCK
1		2 5.0	6929
2		2 8.5	6999
3		2 9.0	6525
4		6 10.5	7771
5		6 6.5	9229
6		6 8.5	799730
7		11 7.0	3917
8		11 9.0	1171
9		15 6.0	2878
10		15 9.0	2410
11		18 7.5	1632
12		18 9.5	3281

Inventory of all products that are the type Bottoms ('B').

```
1372 ✓ SELECT INVENTORY.PROD_NUM, INVENTORY.PROD_SIZE, INVENT_INSTOCK
1373     FROM INVENTORY, PRODUCT
1374     WHERE PROD_CAT = 'B' AND INVENTORY.PROD_NUM = PRODUCT.PROD_NUM AND INVENTORY.PROD_SIZE = PRODUCT.PROD_SIZE;
```

	PROD_NUM	PROD_SIZE	INVENT_INSTOCK
1		1 30	376173
2		1 32	596
3		1 38	98
4		5 30	1307
5		5 40	4774
6		5 44	7408
7		10 28	2245
8		10 34	556
9		14 L	5288
10		14 M	5780
11		20 36	9980

A stored procedure that allows a user to pass in an Invoice ID that shows all product orders associated with the product in the invoice.

```
1376    DELIMITER //
1377 ✓  CREATE PROCEDURE GETORDERINVOICEDETAILS(IN InvoiceID INT)
1378
1379    BEGIN
1380        SELECT
1381            po.ORDER_NUM,
1382            po.ORDER_DATE,
1383            po.ORDER_QTY,
1384            po.ORDER_COST,
1385            po.ORDER_NOTES,
1386            po.VEN_NUM,
1387            i.INVOICE_ID,
1388            i.PURCHASE_DATE,
1389            i.QTY SOLD,
1390            i.TOTAL_PRICE
1391        FROM
1392            PRODUCT_ORDER po
1393        JOIN
1394            INVOICE i ON po.PROD_NUM = i.PROD_NUM
1395                AND po.PROD_SIZE = i.PROD_SIZE
1396        WHERE
1397            i.INVOICE_ID = InvoiceID;
1398    END///
1399    DELIMITER ;
1400    CALL GETORDERINVOICEDETAILS(InvoiceID: 1);
```

	ORDER_NUM	ORDER_DATE	ORDER_QTY	ORDER_COST	ORDER_NOTES		INVOICE_ID	PURCHASE_DATE	QTY SOLD	TOTAL_PRICE
1	1003	1989-09-01	17734	69379.94	Rerum in voluptatem adipisci voluptas accusamus iste quia. Re...		1	1991-10-31	33	329.67
2	1243	2017-09-09	6809	43700.85	Ea qui quae culpa. Vel ratione eum quis molestias eius. Quaer...		1	1991-10-31	33	329.67
3	1483	1974-09-09	29568	75426.38	Id sed corporis bestiae illum labore animi. Nam quia debitis a...		1	1991-10-31	33	329.67
4	1723	1992-04-25	14164	86945.20	Id quia est similique voluptas et culpa repellat. Modi rerum ..		1	1991-10-31	33	329.67
5	1963	1984-09-23	30499	84041.84	Est praesentium hic maxime id et nulla. Doloribus officia quo...		1	1991-10-31	33	329.67

A singular product of Blue Jeans being entered into the product table.

```
443 | INSERT INTO PRODUCT(PROD_NAME, PROD_PRICE, PROD_DESCRIPTION, PROD_CAT, PROD_SIZE, PROD_NUM) VALUES
444 | ('Blue Jeans', 39.99, 'Classic Blue Jeans', 'B', '32', '1');
```

A group of products being entered into the product table.

```
445 | INSERT INTO PRODUCT(PROD_NAME, PROD_PRICE, PROD_DESCRIPTION, PROD_CAT, PROD_SIZE, PROD_NUM) VALUES
446 | ('Canvas Sneakers', 49.99, 'Blue High Top Canvas Sneakers', 'S', '9.0', 2),
447 | ('Cropped T-Shirt', 29.99, 'Basic Cropped Cotton Shirt', 'T', 'S', 3),
448 | ('Fleece Hoodie', 25.99, 'Blue Fleece Jacket', 'T', 'L', 4),
449 | ('Casual Shorts', 19.99, 'Denim High-waisted Shorts', 'B', '30', 5),
450 | ('Beach sandals', 14.99, 'Brown Beach Sandals', 'S', '8.5', 6);
```

A group of invoices being inserted into the invoice table.

```
1241  INSERT INTO INVOICE(CUS_CODE, PROD_NUM, PROD_SIZE, PROD_PRICE, QTY SOLD, PURCHASE_DATE, STORE_ID) VALUES
1242      (3,1,'30',9.99,33,'1991-10-31',3),
1243      (16, 6,'8.5',21.99, 50,'1986-03-05', 16),
1244      (17,6,'8.5',21.99,77,'2006-01-23',17),
1245      (22,8,'L',25.99,12,'2017-07-16',1),
1246      (23,8,'L',27.99,65,'2005-03-24',2),
1247      (24,8,'L',27.99,53,'1997-11-17', 3),
1248      (25,8,'L',27.99,62,'1993-05-15', 4),
1249      (26,9,'L',28.99,97,'1991-11-08',5),
1250      (27,9,'L',28.99,21,'1997-08-28',6),
1251      (34,13,'M',39.99,13,'1989-04-01', 13),
1252      (35,13,'M',39.99,54,'2019-03-07',14),
1253      (40,16,'S',59.99,8,'1983-10-09',19),
1254      (41,16,'S',59.99,95,'2003-01-13',20),
1255      (42,17,'S',59.99,62,'2019-06-29',21),
1256      (43,17,'S',69.99,32,'1978-04-24',1),
1257      (51,1,'30',9.99,51,'1978-04-27',9),
1258      (64,6,'8.5',21.99,39,'2012-02-14',1),
1259      (65,6,'8.5',21.99,11,'1997-12-17',2),
1260      (70,8,'L',25.99,70,'1992-05-08',7),
1261      (71,8,'L',27.99,87,'1980-04-28',8),
1262      (72,8,'L',27.99,95,'1979-02-17',9),
1263      (73,8,'L',27.99,77,'1990-11-08',10),
1264      (74,9,'L',28.99,49,'2007-06-29',11),
1265      (75,9,'L',28.99,36,'1991-12-09',12),
1266      (82,13,'M',39.99,9,'2005-03-31',19),
1267      (83,13,'M',39.99,48,'1976-01-02',20),
1268      (88,16,'S',59.99,17,'1981-12-24',4),
1269      (89,16,'S',59.99,4,'1986-12-01',5),
1270      (90,17,'S',59.99,49,'2009-10-16',6),
1271      (91,17,'S',69.99,83,'2017-05-21',7),
```

Output of a virtual table showing all employees and their stores.

```
1402 CREATE VIEW EMPLOYEE_STORE AS
1403 SELECT
1404     e.EMP_ID,
1405     e.EMP_FNAME,
1406     e.EMP_LNAME,
1407     e.STORE_NUM,
1408     s.STORE_NAME
1409 FROM
1410     EMPLOYEE e
1411 JOIN
1412     STORE s ON e.STORE_NUM = s.STORE_NUM;
1413
1414 ✓ SELECT * FROM EMPLOYEE_STORE;
```

The data dictionary of all the tables in the database.

1416 ✓	SELECT table_name, column_name, column_type, is_nullable, column_comment FROM information_schema.COLUMNS WHERE table_schema = 'kgrassi2693_kgsb_clothing_db_project' ORDER BY table_name, ordinal_position ASC;				
1	BOTTOMS	PROD_NUM	int(11)	NO	
2	BOTTOMS	PROD_SIZE	varchar(6)	NO	
3	BOTTOMS	LENGTH	varchar(50)	YES	
4	BOTTOMS	NUM_OF_POCKETS	int(11)	YES	
5	BOTTOMS	TYPE	varchar(50)	NO	
6	BOTTOMS	WAISTBAND	varchar(50)	YES	
7	BOTTOMS	IS_RIPPED	tinyint(1)	YES	
8	CUSTOMER	CUS_CODE	int(11)	NO	
9	CUSTOMER	CUS_FNAME	varchar(50)	YES	
10	CUSTOMER	CUS_LNAME	varchar(50)	NO	
11	CUSTOMER	CUS_PHONE	bigint(10)	YES	
12	CUSTOMER	CUS_EMAIL	varchar(100)	YES	
13	CUSTOMER	CUS_STATE	varchar(2)	YES	
14	CUSTOMER	CUS_CITY	varchar(100)	NO	
15	CUSTOMER	CUS_ZIP	int(5)	NO	
16	CUSTOMER	CUS_ADDRESS	varchar(100)	NO	
17	EMPLOYEE	EMP_ID	int(11)	NO	
18	EMPLOYEE	EMP_FNAME	varchar(50)	YES	
19	EMPLOYEE	EMP_LNAME	varchar(50)	NO	
20	EMPLOYEE	STORE_NUM	int(11)	NO	
21	EMPLOYEE	EMP_HIREDATE	date	NO	
22	EMPLOYEE	EMP_FIREDATE	date	YES	
23	EMPLOYEE_STORE	EMP_ID	int(11)	NO	
24	EMPLOYEE_STORE	EMP_FNAME	varchar(50)	YES	
25	EMPLOYEE_STORE	EMP_LNAME	varchar(50)	NO	
26	EMPLOYEE_STORE	STORE_NUM	int(11)	NO	
27	EMPLOYEE_STORE	STORE_NAME	varchar(50)	NO	
28	INVENTORY	PROD_NUM	int(11)	NO	
29	INVENTORY	PROD_SIZE	varchar(6)	NO	
30	INVENTORY	INVENT_INSTOCK	int(11)	YES	
31	INVENTORY	INV_NOTES	varchar(100)	YES	
32	INVOICE	INVOICE_ID	int(11)	NO	
33	INVOICE	CUS_CODE	int(11)	NO	
34	INVOICE	PROD_NUM	int(11)	NO	
35	INVOICE	PROD_SIZE	varchar(6)	NO	
36	INVOICE	PROD_PRICE	decimal(10,2)	NO	
37	INVOICE	QTY_SOLD	int(11)	NO	
38	INVOICE	PURCHASE_DATE	date	NO	
39	INVOICE	TOTAL_PRICE	decimal(12,2)	YES	
40	INVOICE	STORE_ID	int(11)	NO	
41	INVOICE_HISTORY	INV_HIST_ID	int(11)	NO	
42	INVOICE_HISTORY	CUS_CODE	int(11)	NO	
43	INVOICE_HISTORY	TOTAL_QTY	int(11)	NO	
44	INVOICE_HISTORY	TOTAL_PRICE	decimal(10,0)	NO	
45	PRODUCT	PROD_NUM	int(11)	NO	
46	PRODUCT	PROD_NAME	varchar(50)	NO	
47	PRODUCT	PROD_PRICE	decimal(10,2)	NO	
48	PRODUCT	PROD_DESCRIPTION	text	YES	
49	PRODUCT	PROD_CAT	varchar(1)	NO	
50	PRODUCT	PROD_SIZE	varchar(6)	NO	
51	PRODUCT_ORDER	PROD_NUM	int(11)	NO	
52	PRODUCT_ORDER	PROD_SIZE	varchar(6)	NO	
53	PRODUCT_ORDER	VEN_NUM	int(11)	NO	
54	PRODUCT_ORDER	ORDER_NUM	int(11)	NO	
55	PRODUCT_ORDER	ORDER_DATE	date	NO	
56	PRODUCT_ORDER	ORDER_QTY	int(11)	NO	
57	PRODUCT_ORDER	ORDER_COST	decimal(12,2)	NO	
58	PRODUCT_ORDER	ORDER_NOTES	varchar(100)	YES	
59	PRODUCT_VENDOR	VENDOR_ID	int(11)	NO	
60	PRODUCT_VENDOR	VEN_NAME	varchar(50)	NO	
61	PRODUCT_VENDOR	VEN_COUNTRY	varchar(50)	NO	
62	PRODUCT_VENDOR	VEN_STATE	varchar(2)	YES	
63	PRODUCT_VENDOR	VEN_CITY	varchar(100)	NO	
64	PRODUCT_VENDOR	VEN_ADDRESS	varchar(50)	NO	
65	PRODUCT_VENDOR	VEN_TOTAL_BILLED	decimal(12,2)	YES	
66	PRODUCT_VENDOR	VEN_TOTAL_PRODUCED	int(11)	YES	
67	REVIEW	REVIEW_ID	int(11)	NO	
68	REVIEW	PROD_NUM	int(11)	NO	
69	REVIEW	PROD_SIZE	varchar(6)	NO	
70	REVIEW	CUS_CODE	int(11)	NO	
71	REVIEW	RATING	int(1)	NO	
72	REVIEW	REVIEW_DATE	date	NO	
73	REVIEW	COMMENTS	text	YES	
74	SHOES	PROD_NUM	int(11)	NO	
75	SHOES	PROD_SIZE	varchar(6)	NO	
76	SHOES	HEEL_HEIGHT	varchar(10)	YES	
77	SHOES	SHOE_TYPE	varchar(20)	NO	
78	SHOES	FASTENINGS	varchar(20)	NO	
79	SHOES	WIDTH	varchar(2)	NO	
80	SHOES	ARCH_SUPPORT	varchar(10)	YES	
81	STORE	STORE_NUM	int(11)	NO	
82	STORE	STORE_NAME	varchar(50)	NO	
83	STORE	STORE_STATE	varchar(2)	YES	
84	STORE	STORE_ZIP	int(5)	NO	
85	STORE	STORE_CITY	varchar(50)	NO	
86	STORE	STORE_ADDRESS	varchar(50)	NO	
87	STORE_INVENTORY	STORE_INVENT_NUM	int(11)	NO	
88	STORE_INVENTORY	PROD_NUM	int(11)	NO	
89	STORE_INVENTORY	STORE_NUM	int(11)	NO	
90	STORE_INVENTORY	INVENT_INSTOCK	int(11)	YES	
91	STORE_INVENTORY	PROD_SIZE	varchar(6)	NO	
92	TOPS	PROD_NUM	int(11)	NO	
93	TOPS	PROD_SIZE	varchar(6)	NO	
94	TOPS	SLEEVE_LENGTH	varchar(25)	NO	
95	TOPS	COLLAR_TYPE	varchar(50)	NO	
96	TOPS	IS_CROPPED	tinyint(1)	YES	
97	TOPS	DESIGN	varchar(100)	NO	

For KGSB Clothing, we proposed a database that focused on improving inventory management, customer information tracking, and ensuring data integrity. We implemented a robust database system that adds inventory management processes, accurate customer records, and enhanced data accessibility.

Our database solution has various key components, including a centralized product catalog, real-time inventory tracking, detailed customer profiles, order processing mechanisms, and comprehensive reporting functionalities. The product catalog provides detailed information about each product and facilitates better inventory management and order processing. Real-time inventory tracking will ensure accurate stock levels, enabling proactive restocking and efficient handling of returns. Furthermore, the database solution prioritizes data integrity and security, ensuring that all information remains consistent and accurate. Overall, the proposed database solution will empower KGSB Clothing to optimize their operations, improve customer satisfaction, and drive sustainable growth in the competitive retail market.

In conclusion, the database solution for KGSB Clothing offers a comprehensive approach to addressing their inventory and data management challenges. With our database, KGSB Clothing can streamline operations, enhance data accuracy and unlock new opportunities for business growth and success. With a focus on efficiency, reliability, and data security, the proposed solution aligns with KGSB Clothing's objectives of optimizing operations and delivering exceptional customer experiences in the ever-evolving retail landscape.

	Knoelle Grassi	Samatha Bentley
Editing & Formatting		x
Typing the Database	x	
Creating the PowerPoint		x
Creating the EERD	x	x
Page 1	x	
Page 2		x
Page 3	x	
Page 4-13	x	
Creating the Data for the Database	x	x