Harshika Santoshi

TABLE SCHEMAS

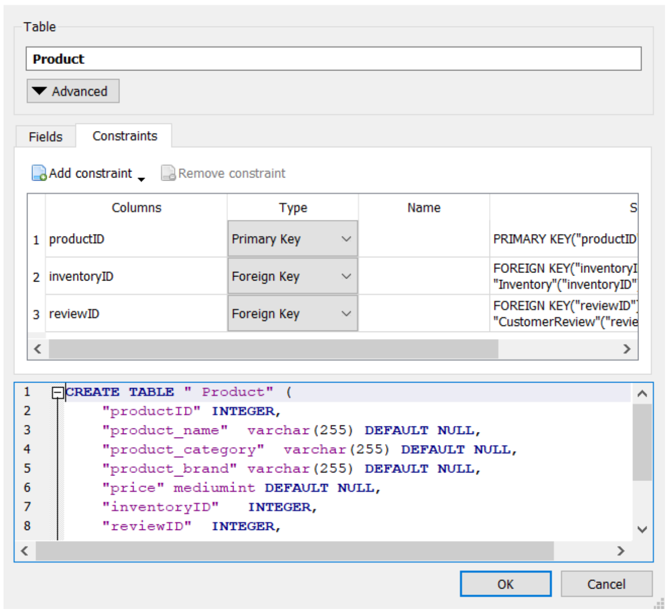


Table: Product

	Columns	Type	Name	S
1	productID	Primary Key		PRIMARY KEY("productID
2	inventoryID	Foreign Key		FOREIGN KEY("inventoryI "Inventory"("inventoryID"
3	reviewID	Foreign Key		FOREIGN KEY("reviewID" "CustomerReview"("revie

```
CREATE TABLE " Product" (
    "productID" INTEGER,
    "product_name"  varchar(255) DEFAULT NULL,
    "product_category"  varchar(255) DEFAULT NULL,
    "product_brand" varchar(255) DEFAULT NULL,
    "price" mediumint DEFAULT NULL,
    "inventoryID"   INTEGER,
    "reviewID"  INTEGER,
```

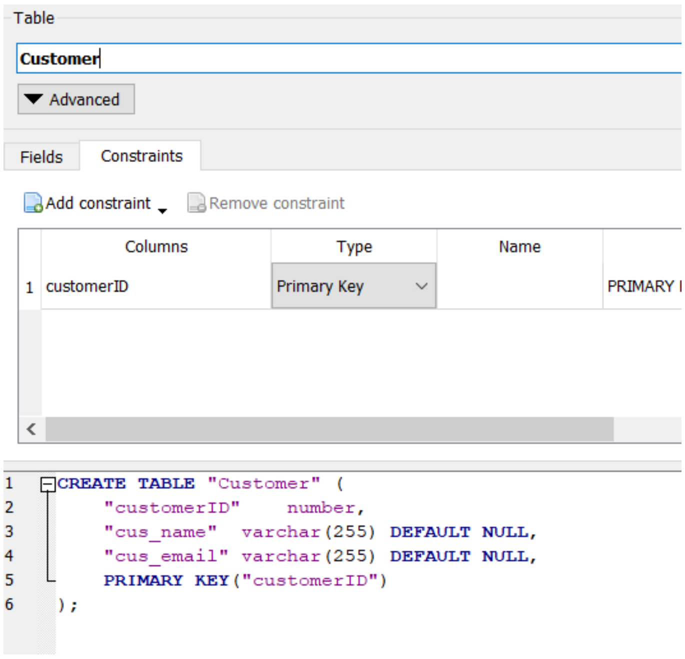
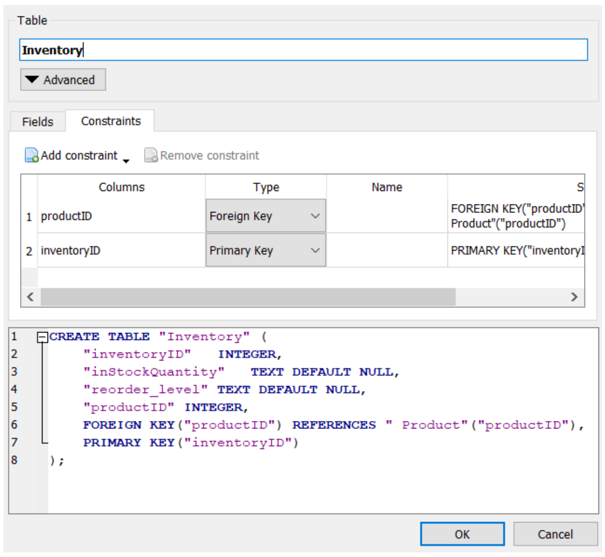


Table: Customer

	Columns	Type	Name	
1	customerID	Primary Key		PRIMARY I

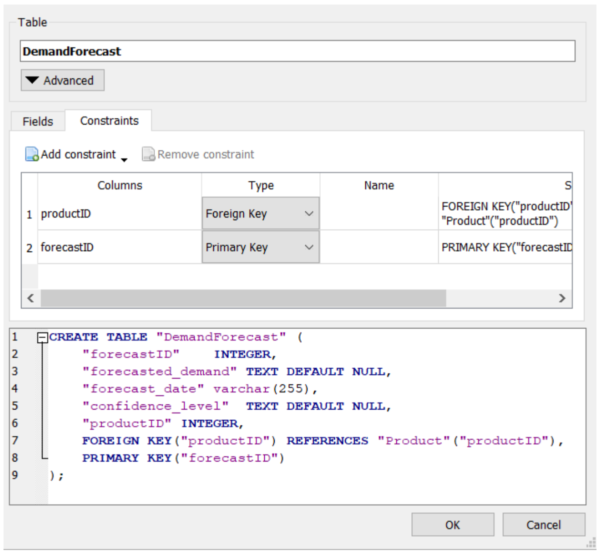
```
CREATE TABLE "Customer"  (
    "customerID"    number,
    "cus_name"   varchar(255) DEFAULT NULL,
    "cus_email"  varchar(255) DEFAULT NULL,
    PRIMARY KEY("customerID")
);
```

Harshika Santoshi



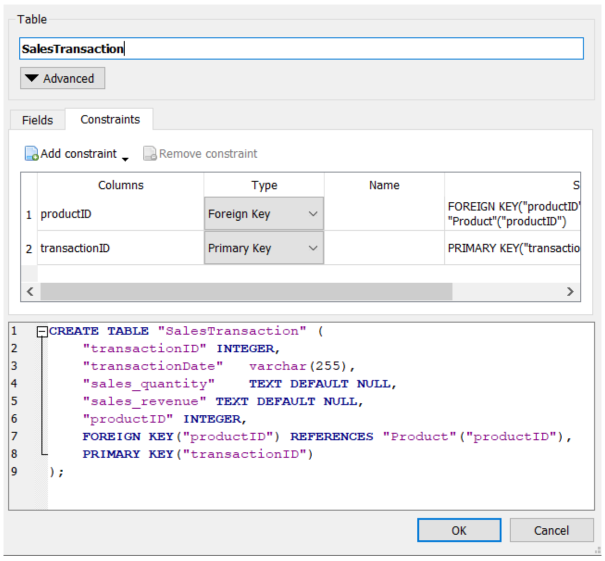
	Columns	Type	Name	S
1	productID	Foreign Key		FOREIGN KEY("productID" Product"("productID")
2	inventoryID	Primary Key		PRIMARY KEY("inventoryI

```
CREATE TABLE "Inventory" (
    "inventoryID"   INTEGER,
    "inStockQuantity"   TEXT DEFAULT NULL,
    "reorder_level" TEXT DEFAULT NULL,
    "productID" INTEGER,
    FOREIGN KEY("productID") REFERENCES " Product"("productID"),
    PRIMARY KEY("inventoryID")
);
```



	Columns	Type	Name	S
1	productID	Foreign Key		FOREIGN KEY("productID" "Product"("productID")
2	forecastID	Primary Key		PRIMARY KEY("forecastID

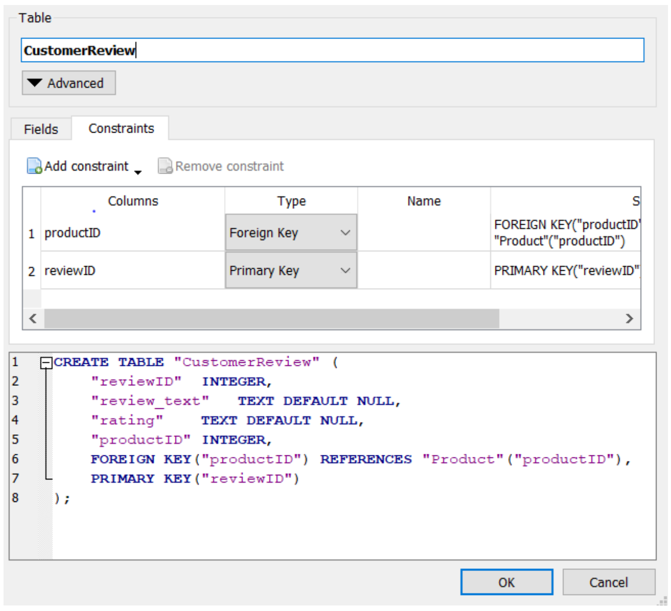
```
CREATE TABLE "DemandForecast" (
    "forecastID"    INTEGER,
    "forecasted_demand" TEXT DEFAULT NULL,
    "forecast_date" varchar(255),
    "confidence_level"  TEXT DEFAULT NULL,
    "productID" INTEGER,
    FOREIGN KEY("productID") REFERENCES "Product"("productID"),
    PRIMARY KEY("forecastID")
);
```



	Columns	Type	Name	S
1	productID	Foreign Key		FOREIGN KEY("productID" "Product"("productID")
2	transactionID	Primary Key		PRIMARY KEY("transactio

```
CREATE TABLE "SalesTransaction" (
    "transactionID" INTEGER,
    "transactionDate"   varchar(255),
    "sales_quantity"    TEXT DEFAULT NULL,
    "sales_revenue" TEXT DEFAULT NULL,
    "productID" INTEGER,
    FOREIGN KEY("productID") REFERENCES "Product"("productID"),
    PRIMARY KEY("transactionID")
);
```

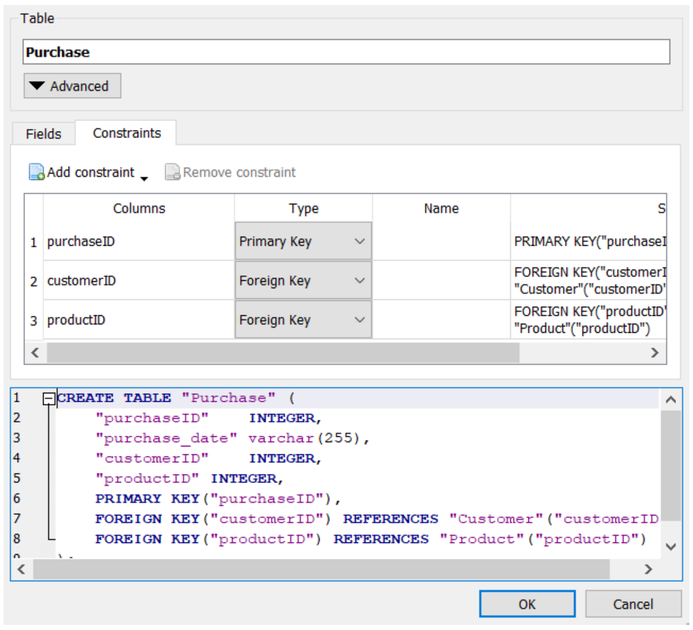
Harshika Santoshi



CustomerReview

	Columns	Type	Name	S
1	productID	Foreign Key		FOREIGN KEY("productID" "Product"("productID")
2	reviewID	Primary Key		PRIMARY KEY("reviewID"

```
CREATE TABLE "CustomerReview" (
    "reviewID"  INTEGER,
    "review_text"   TEXT DEFAULT NULL,
    "rating"    TEXT DEFAULT NULL,
    "productID" INTEGER,
    FOREIGN KEY("productID") REFERENCES "Product"("productID"),
    PRIMARY KEY("reviewID")
);
```



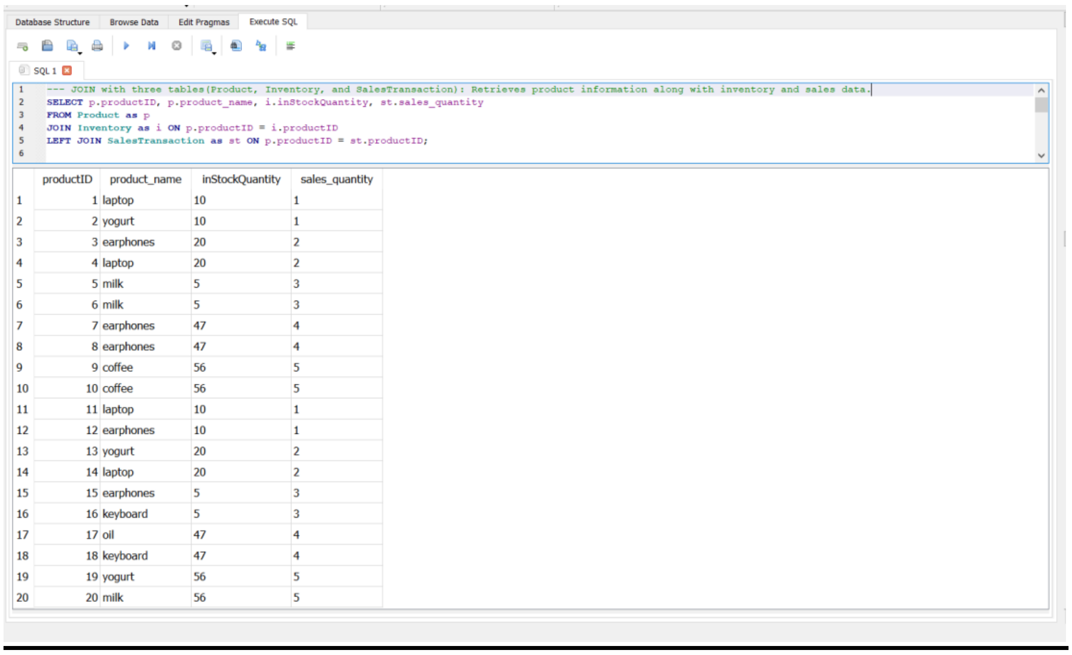
Purchase

	Columns	Type	Name	S
1	purchaseID	Primary Key		PRIMARY KEY("purchaseI
2	customerID	Foreign Key		FOREIGN KEY("customerI "Customer"("customerID"
3	productID	Foreign Key		FOREIGN KEY("productID" "Product"("productID")

```
CREATE TABLE "Purchase" (
    "purchaseID"    INTEGER,
    "purchase_date" varchar(255),
    "customerID"    INTEGER,
    "productID" INTEGER,
    PRIMARY KEY("purchaseID"),
    FOREIGN KEY("customerID") REFERENCES "Customer"("customerID
    FOREIGN KEY("productID") REFERENCES "Product"("productID")
```

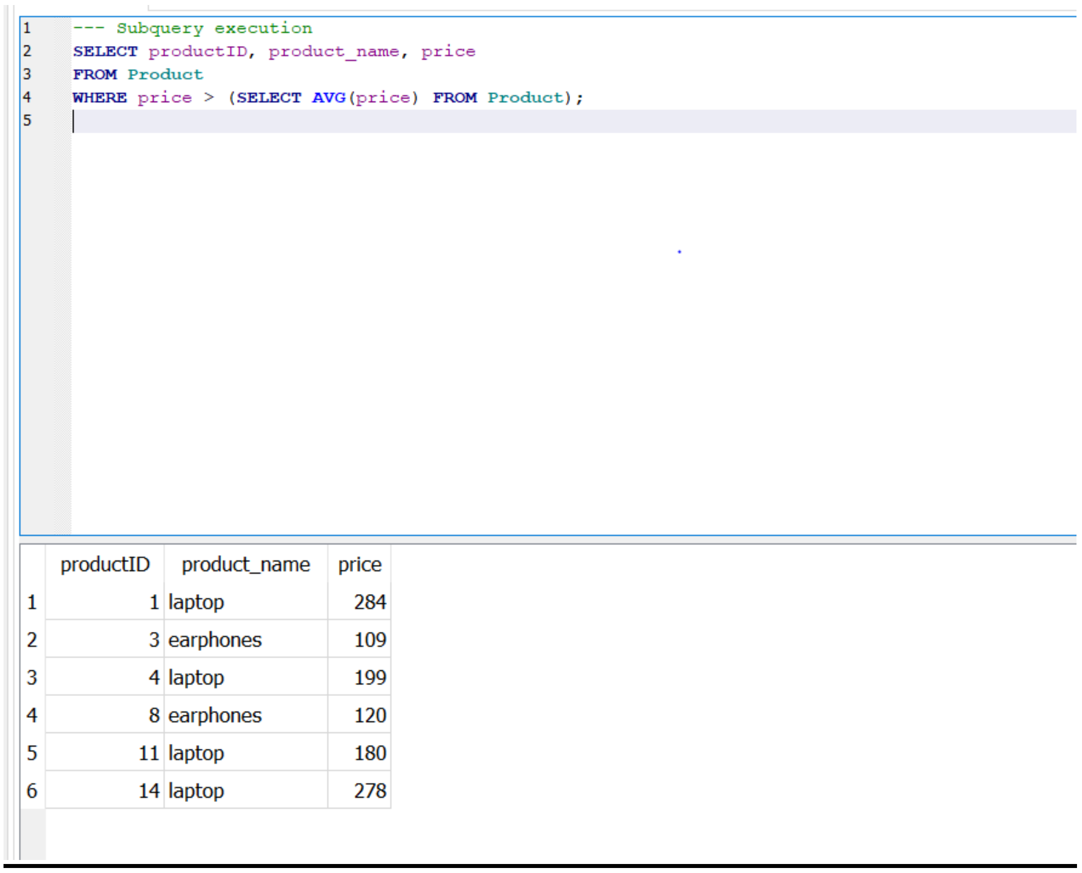
Harshika Santoshi

QUERY EXECUTIONS



```sql
--- JOIN with three tables(Product, Inventory, and SalesTransaction): Retrieves product information along with inventory and sales data.
SELECT p.productID, p.product_name, i.inStockQuantity, st.sales_quantity
FROM Product as p
JOIN Inventory as i ON p.productID = i.productID
LEFT JOIN SalesTransaction as st ON p.productID = st.productID;
```

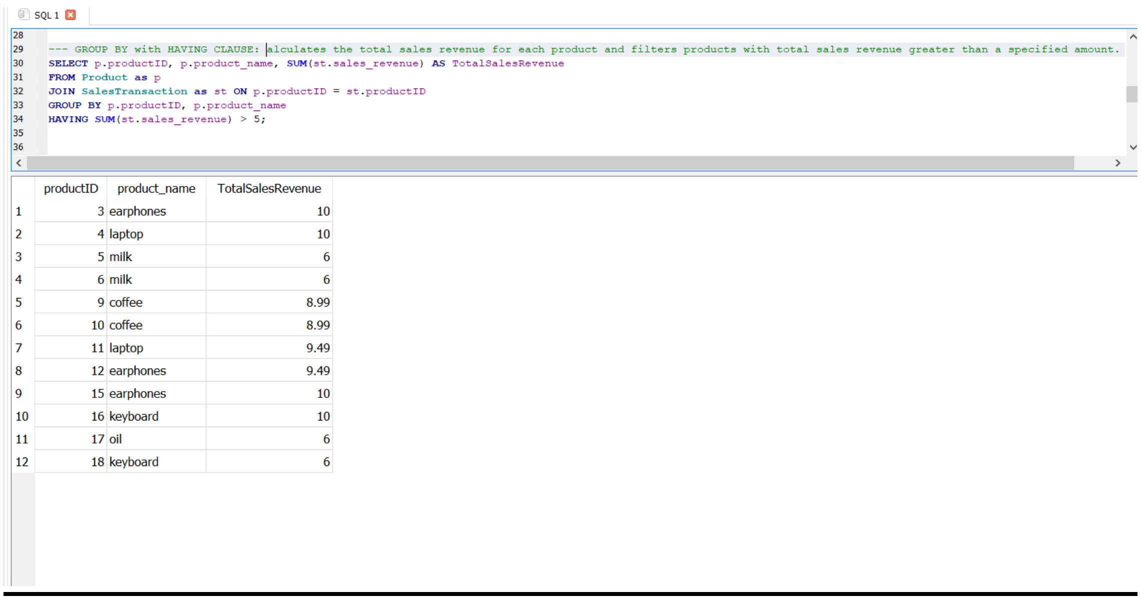
	productID	product_name	inStockQuantity	sales_quantity
1	1	laptop	10	1
2	2	yogurt	10	1
3	3	earphones	20	2
4	4	laptop	20	2
5	5	milk	5	3
6	6	milk	5	3
7	7	earphones	47	4
8	8	earphones	47	4
9	9	coffee	56	5
10	10	coffee	56	5
11	11	laptop	10	1
12	12	earphones	10	1
13	13	yogurt	20	2
14	14	laptop	20	2
15	15	earphones	5	3
16	16	keyboard	5	3
17	17	oil	47	4
18	18	keyboard	47	4
19	19	yogurt	56	5
20	20	milk	56	5



```sql
--- Subquery execution
SELECT productID, product_name, price
FROM Product
WHERE price > (SELECT AVG(price) FROM Product);
```

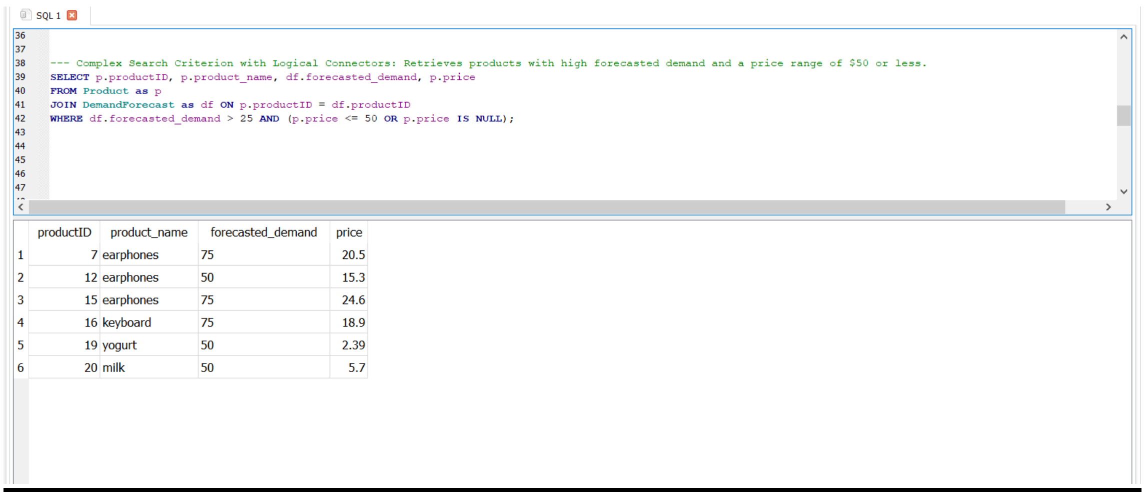
	productID	product_name	price
1	1	laptop	284
2	3	earphones	109
3	4	laptop	199
4	8	earphones	120
5	11	laptop	180
6	14	laptop	278

Harshika Santoshi



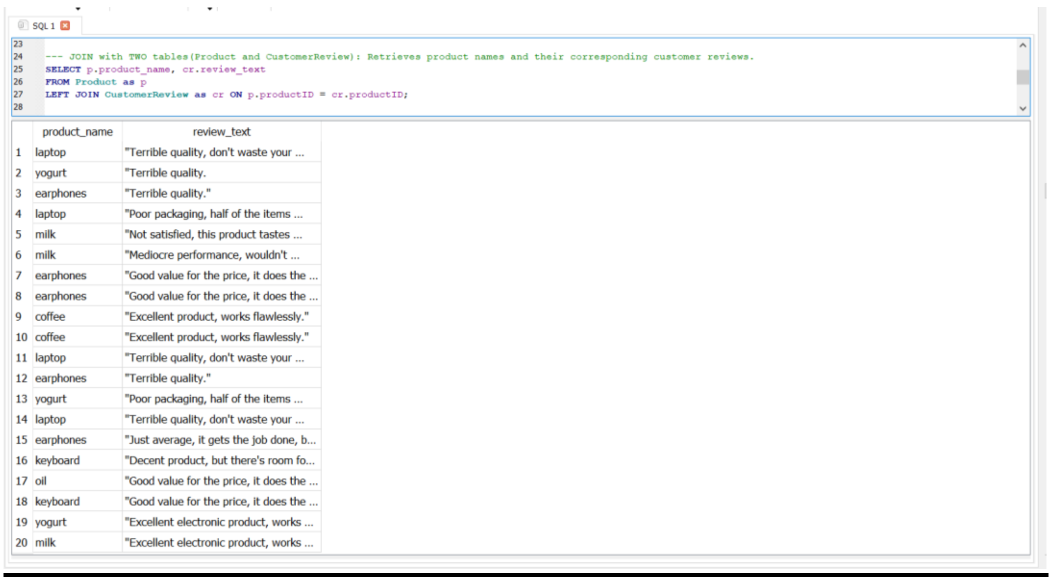
```sql
--- GROUP BY with HAVING CLAUSE: alculates the total sales revenue for each product and filters products with total sales revenue greater than a specified amount.
SELECT p.productID, p.product_name, SUM(st.sales_revenue) AS TotalSalesRevenue
FROM Product as p
JOIN SalesTransaction as st ON p.productID = st.productID
GROUP BY p.productID, p.product_name
HAVING SUM(st.sales_revenue) > 5;
```

	productID	product_name	TotalSalesRevenue
1	3	earphones	10
2	4	laptop	10
3	5	milk	6
4	6	milk	6
5	9	coffee	8.99
6	10	coffee	8.99
7	11	laptop	9.49
8	12	earphones	9.49
9	15	earphones	10
10	16	keyboard	10
11	17	oil	6
12	18	keyboard	6



```sql
--- Complex Search Criterion with Logical Connectors: Retrieves products with high forecasted demand and a price range of $50 or less.
SELECT p.productID, p.product_name, df.forecasted_demand, p.price
FROM Product as p
JOIN DemandForecast as df ON p.productID = df.productID
WHERE df.forecasted_demand > 25 AND (p.price <= 50 OR p.price IS NULL);
```

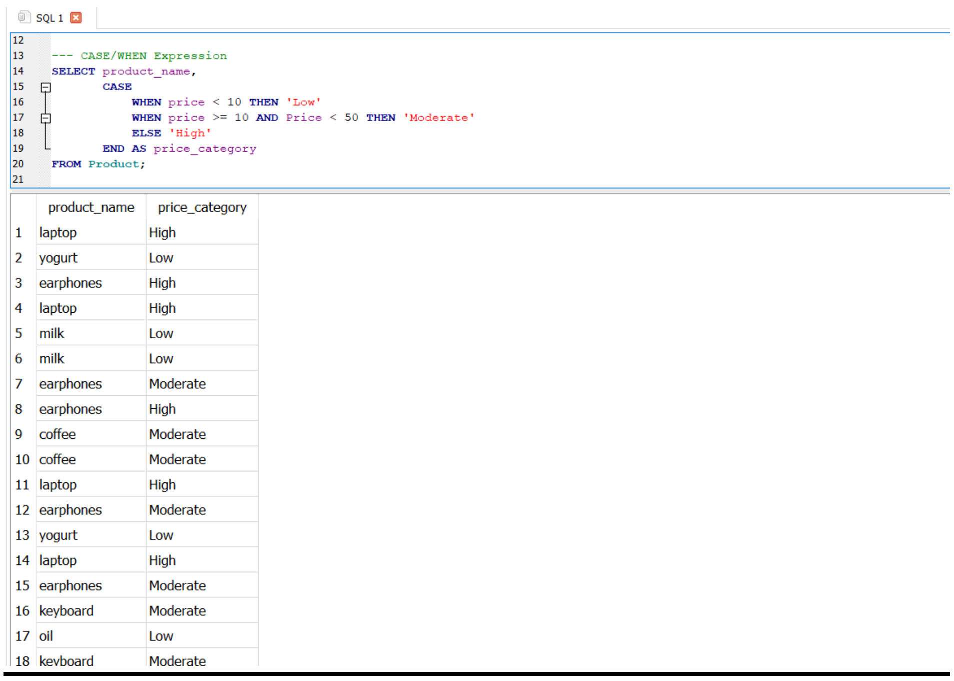
	productID	product_name	forecasted_demand	price
1	7	earphones	75	20.5
2	12	earphones	50	15.3
3	15	earphones	75	24.6
4	16	keyboard	75	18.9
5	19	yogurt	50	2.39
6	20	milk	50	5.7



```sql
--- JOIN with TWO tables(Product and CustomerReview): Retrieves product names and their corresponding customer reviews.
SELECT p.product_name, cr.review_text
FROM Product as p
LEFT JOIN CustomerReview as cr ON p.productID = cr.productID;
```

	product_name	review_text
1	laptop	"Terrible quality, don't waste your ...
2	yogurt	"Terrible quality.
3	earphones	"Terrible quality."
4	laptop	"Poor packaging, half of the items ...
5	milk	"Not satisfied, this product tastes ...
6	milk	"Mediocre performance, wouldn't ...
7	earphones	"Good value for the price, it does the ...
8	earphones	"Good value for the price, it does the ...
9	coffee	"Excellent product, works flawlessly."
10	coffee	"Excellent product, works flawlessly."
11	laptop	"Terrible quality, don't waste your ...
12	earphones	"Terrible quality."
13	yogurt	"Poor packaging, half of the items ...
14	laptop	"Terrible quality, don't waste your ...
15	earphones	"Just average, it gets the job done, b...
16	keyboard	"Decent product, but there's room fo...
17	oil	"Good value for the price, it does the ...
18	keyboard	"Good value for the price, it does the ...
19	yogurt	"Excellent electronic product, works ...
20	milk	"Excellent electronic product, works ...

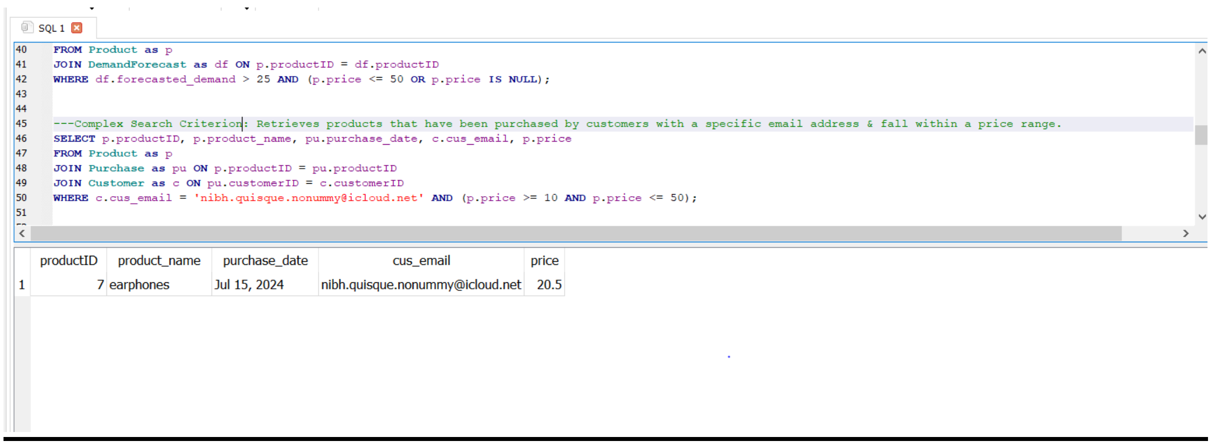
Harshika Santoshi



SQL 1

```
--- CASE/WHEN Expression
SELECT product_name,
       CASE
           WHEN price < 10 THEN 'Low'
           WHEN price >= 10 AND Price < 50 THEN 'Moderate'
           ELSE 'High'
       END AS price_category
FROM Product;
```

	product_name	price_category
1	laptop	High
2	yogurt	Low
3	earphones	High
4	laptop	High
5	milk	Low
6	milk	Low
7	earphones	Moderate
8	earphones	High
9	coffee	Moderate
10	coffee	Moderate
11	laptop	High
12	earphones	Moderate
13	yogurt	Low
14	laptop	High
15	earphones	Moderate
16	keyboard	Moderate
17	oil	Low
18	keyboard	Moderate



SQL 1

```
FROM Product as p
JOIN DemandForecast as df ON p.productID = df.productID
WHERE df.forecasted_demand > 25 AND (p.price <= 50 OR p.price IS NULL);


---Complex Search Criterion: Retrieves products that have been purchased by customers with a specific email address & fall within a price range.
SELECT p.productID, p.product_name, pu.purchase_date, c.cus_email, p.price
FROM Product as p
JOIN Purchase as pu ON p.productID = pu.productID
JOIN Customer as c ON pu.customerID = c.customerID
WHERE c.cus_email = 'nibh.quisque.nonummy@icloud.net' AND (p.price >= 10 AND p.price <= 50);
```

	productID	product_name	purchase_date	cus_email	price
1	7	earphones	Jul 15, 2024	nibh.quisque.nonummy@icloud.net	20.5