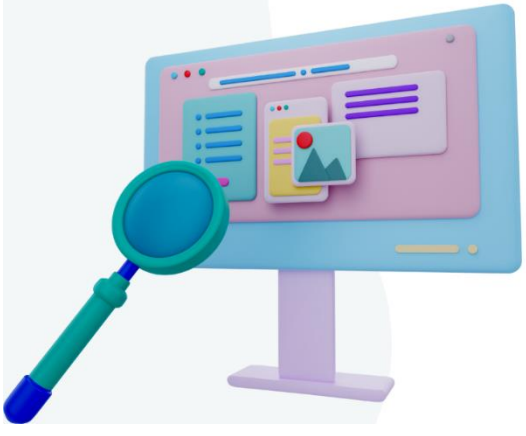
Inventory Database & Data Eploration with SQL-Query

Sagar Limbachiya

DATABASE CREATION & ANALYSIS OF INSTOCK INVENTORY



ERD – Showing Table Relations

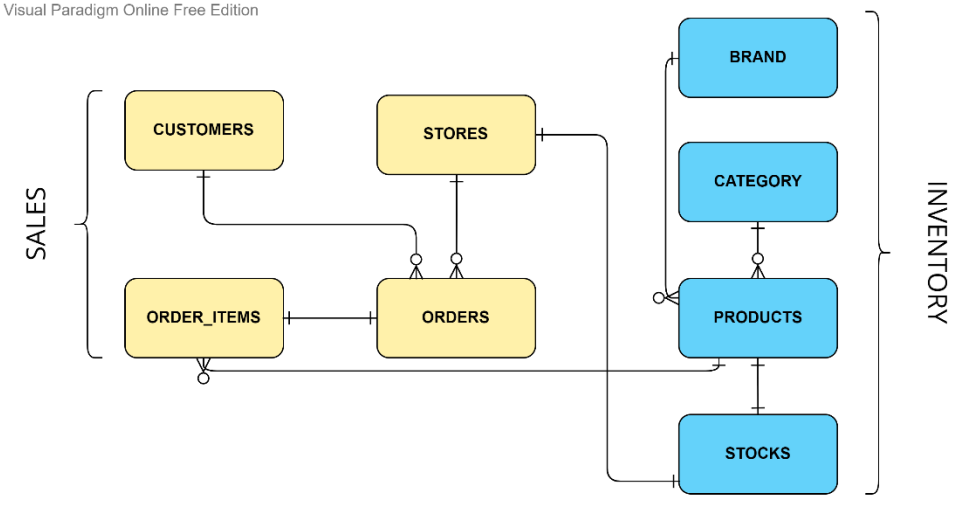


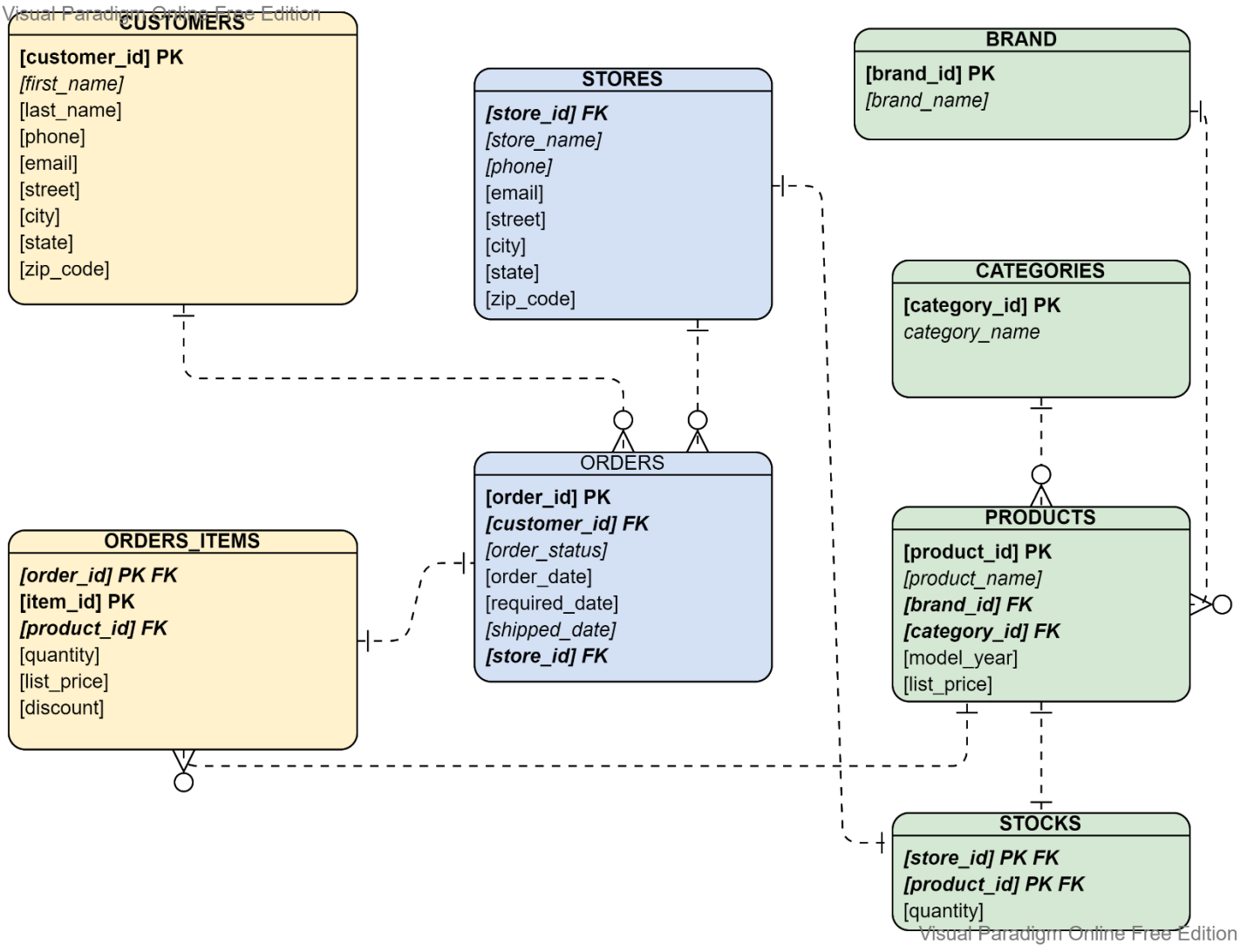
Table Relationships

1.Brand & Product tables will have **one to many relation**
2.Category & Product table will have **one to many relation**

3.Product & Stock table will have **one to one relation** only with product_id as Foreign Key
4.Orders & Order items will have **one to one relationship** with order_id being Foreign Key
5.Stores & Orders will have **one to many relations** as store could have multiple orders
6.Customer Table & Order table will have **one to many relationships**

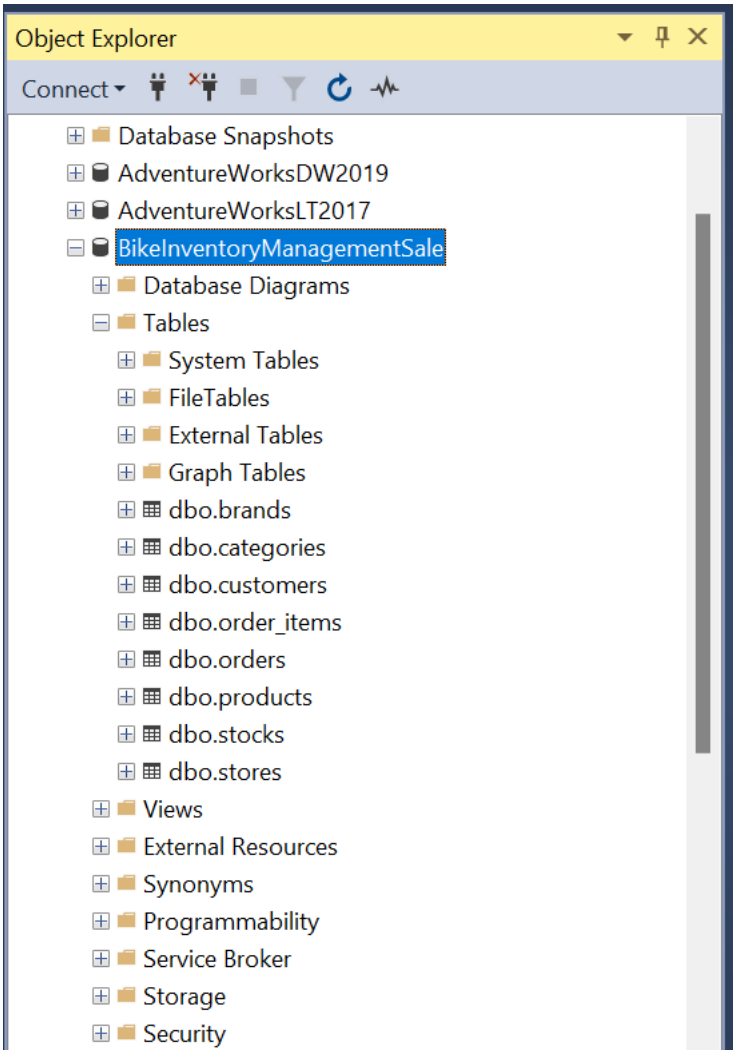
With the entities and relationships identified in Part B, identify suitable attributes for each entity with the respective Primary and Foreign Keys. Create a detailed Entity Relationship Diagram with the findings.

ERD – Showing detailed table-column relationships



D) With reference to the detailed entity relationship diagram in Part C, complete the below tasks.

- **Create a database in SQL Server (with SQL).**

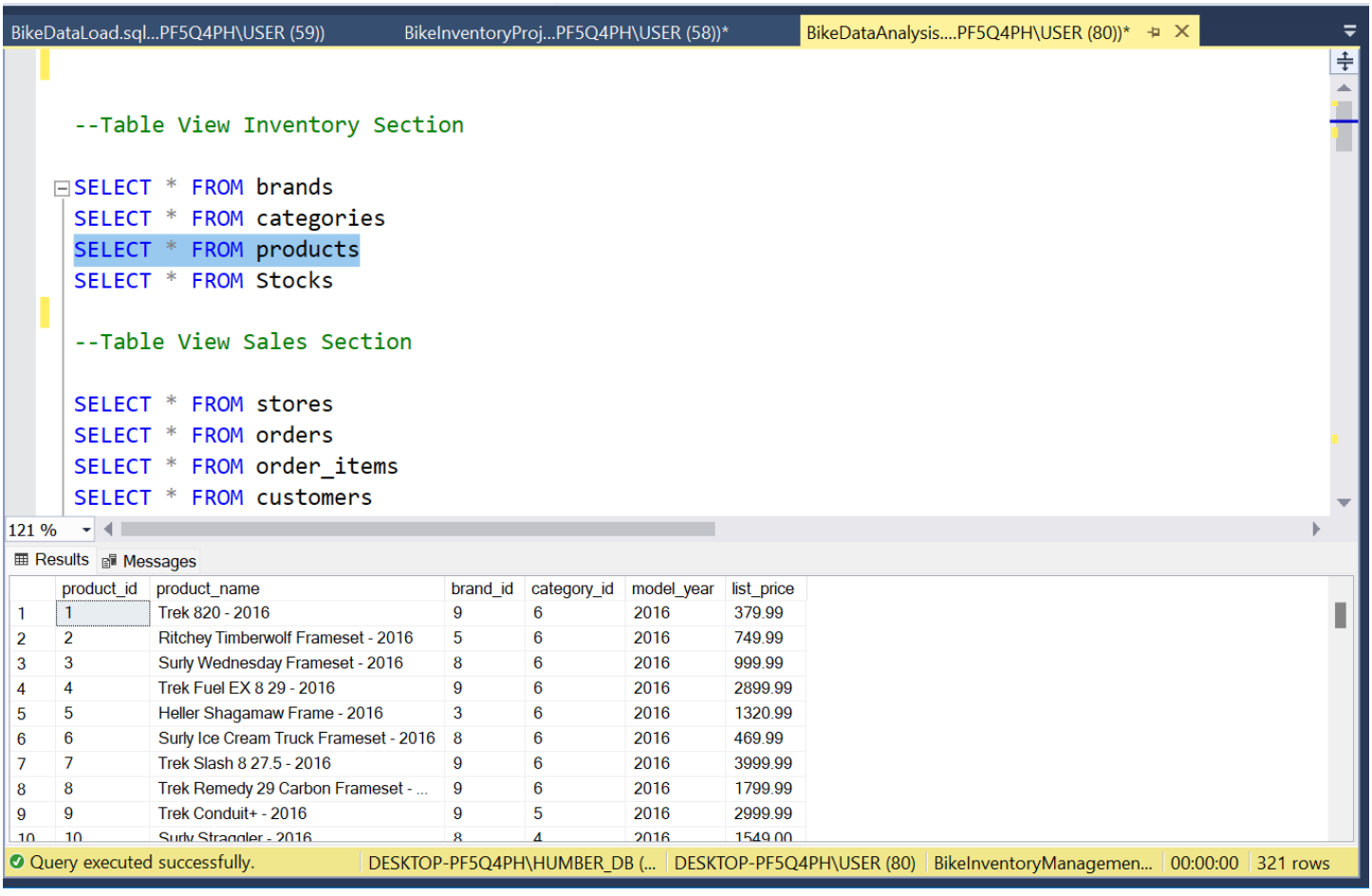


• Create the tables identified, in the SQL Server database with respective constraints (with SQL).

```
--Table 1 Inventory Section

--Category Table-1
```

```sql
CREATE TABLE categories (
	category_id INT IDENTITY (1, 1) PRIMARY KEY,
	category_name VARCHAR (255) NOT NULL
);

--Product Brand Table-2

CREATE TABLE brands (
	brand_id INT IDENTITY (1, 1) PRIMARY KEY,
	brand_name VARCHAR (255) NOT NULL
);


--Products available Table-3

CREATE TABLE products (
	product_id INT IDENTITY (1, 1) PRIMARY KEY,
	product_name VARCHAR (255) NOT NULL,
	brand_id INT NOT NULL,
	category_id INT NOT NULL,
	model_year SMALLINT NOT NULL,
	list_price DECIMAL (10, 2) NOT NULL,
	FOREIGN KEY (category_id) REFERENCES categories (category_id) ,
	FOREIGN KEY (brand_id) REFERENCES brands (brand_id)
);

--Quantity Available Table-4

CREATE TABLE stocks (
	store_id INT,
	product_id INT,
	quantity INT,
	PRIMARY KEY (store_id, product_id),
	FOREIGN KEY (store_id) REFERENCES stores (store_id) ,
	FOREIGN KEY (product_id) REFERENCES products (product_id)
);









--Inventor Sales Section

--Customer Sales Table-1

CREATE TABLE customers (
	customer_id INT IDENTITY (1, 1) PRIMARY KEY,
	first_name VARCHAR (255) NOT NULL,
	last_name VARCHAR (255) NOT NULL,
	phone VARCHAR (25),
```

```
      email VARCHAR (255) NOT NULL,
      street VARCHAR (255),
      city VARCHAR (50),
      state VARCHAR (25),
      zip_code VARCHAR (5)
);

--Store Sales Table-2

CREATE TABLE stores (
      store_id INT IDENTITY (1, 1) PRIMARY KEY,
      store_name VARCHAR (255) NOT NULL,
      phone VARCHAR (25),
      email VARCHAR (255),
      street VARCHAR (255),
      city VARCHAR (255),
      state VARCHAR (10),
      zip_code VARCHAR (5)
);

--Product Orders Table-3

CREATE TABLE orders (
      order_id INT IDENTITY (1, 1) PRIMARY KEY,
      customer_id INT,
      order_status tinyint NOT NULL,
      -- Order status: 1 = Pending; 2 = Processing; 3 = Rejected; 4 = Completed
      order_date DATE NOT NULL,
      required_date DATE NOT NULL,
      shipped_date DATE,
      store_id INT NOT NULL,
      FOREIGN KEY (customer_id) REFERENCES customers (customer_id) ,
      FOREIGN KEY (store_id) REFERENCES stores (store_id) ,
);

--Order Sales Table-4

CREATE TABLE order_items (
      order_id INT,
      item_id INT,
      product_id INT NOT NULL,
      quantity INT NOT NULL,
      list_price DECIMAL (10, 2) NOT NULL,
      discount DECIMAL (4, 2) NOT NULL DEFAULT 0,
      PRIMARY KEY (order_id, item_id),
      FOREIGN KEY (order_id) REFERENCES orders (order_id) ,
      FOREIGN KEY (product_id) REFERENCES products (product_id)
);
```



• Populate each table with dummy data (with SQL).

```
---Inserting Bulk Data

GO

-- import the file--brands.csv
```

```
BULK INSERT dbo.brands
FROM 'C:\Users\USER\Dropbox\PC\Desktop\sql
project\bikestore_Inventory_Management\brands.csv'
WITH
(
        FORMAT='CSV',
        FIRSTROW=1
)
GO

-- import the file--product_categories.csv
BULK INSERT dbo.categories
FROM 'C:\Users\USER\Dropbox\PC\Desktop\sql
project\bikestore_Inventory_Management\product_categories.csv'
WITH
(
        FORMAT='CSV',
        FIRSTROW=1
)
GO

-- import the file--sales_customers.csv
BULK INSERT dbo.customers
FROM 'C:\Users\USER\Dropbox\PC\Desktop\sql
project\bikestore_Inventory_Management\sales_customers.csv'
WITH
(
        FORMAT='CSV',
        FIRSTROW=1
)
GO

-- import the file--sales_orders_items.csv
BULK INSERT dbo.order_items
FROM 'C:\Users\USER\Dropbox\PC\Desktop\sql
project\bikestore_Inventory_Management\sales_orders_items.csv'
WITH
(
        FORMAT='CSV',
        FIRSTROW=1
)
GO

-- import the file--sales_orders.csv
BULK INSERT dbo.orders
FROM 'C:\Users\USER\Dropbox\PC\Desktop\sql
project\bikestore_Inventory_Management\sales_orders.csv'
WITH
(
        FORMAT='CSV',
        FIRSTROW=1
)
GO
```

```
-- import the file--products.csv
BULK INSERT dbo.products
FROM 'C:\Users\USER\Dropbox\PC\Desktop\sql
project\bikestore_Inventory_Management\products.csv'
WITH
(
        FORMAT='CSV',
        FIRSTROW=1
)
GO

-- import the file--sales_stores.csv
BULK INSERT dbo.stores
FROM 'C:\Users\USER\Dropbox\PC\Desktop\sql
project\bikestore_Inventory_Management\sales_stores.csv'
WITH
(
        FORMAT='CSV',
        FIRSTROW=1
)
GO
```

1.) Query-1

```
--**Using subquery & Left Join, Full Join to get Inventory Price then get
Inventory Value
--& store details --Query uses 2 tables to get below data
```

```sql
select x.store_id,y.store_name,y.city,x.Quantity,x.Inventory_Value_Millions from
(
SELECT store_id,sum(s.quantity) as Quantity,sum(p.list_price*s.quantity/1000000)
as Inventory_Value_Millions FROM
Stocks s left join products p on p.product_id=s.product_id
group by store_id
) x join stores y on x.store_id=y.store_id
order by 1
```

screenshot

```sql
--**Using subquery & Left Join, Full Join to get Inventory Price then get Inventory Value
--& store details --Query uses 2 tables to get below data

select x.store_id,y.store_name,y.city,x.Quantity,x.Inventory_Value_Millions from (
SELECT store_id,sum(s.quantity) as Quantity,sum(p.list_price*s.quantity/1000000)
as Inventory_Value_Millions FROM
Stocks s left join products p on p.product_id=s.product_id
group by store_id
) x join stores y on x.store_id=y.store_id
order by 1
```

esults Messages

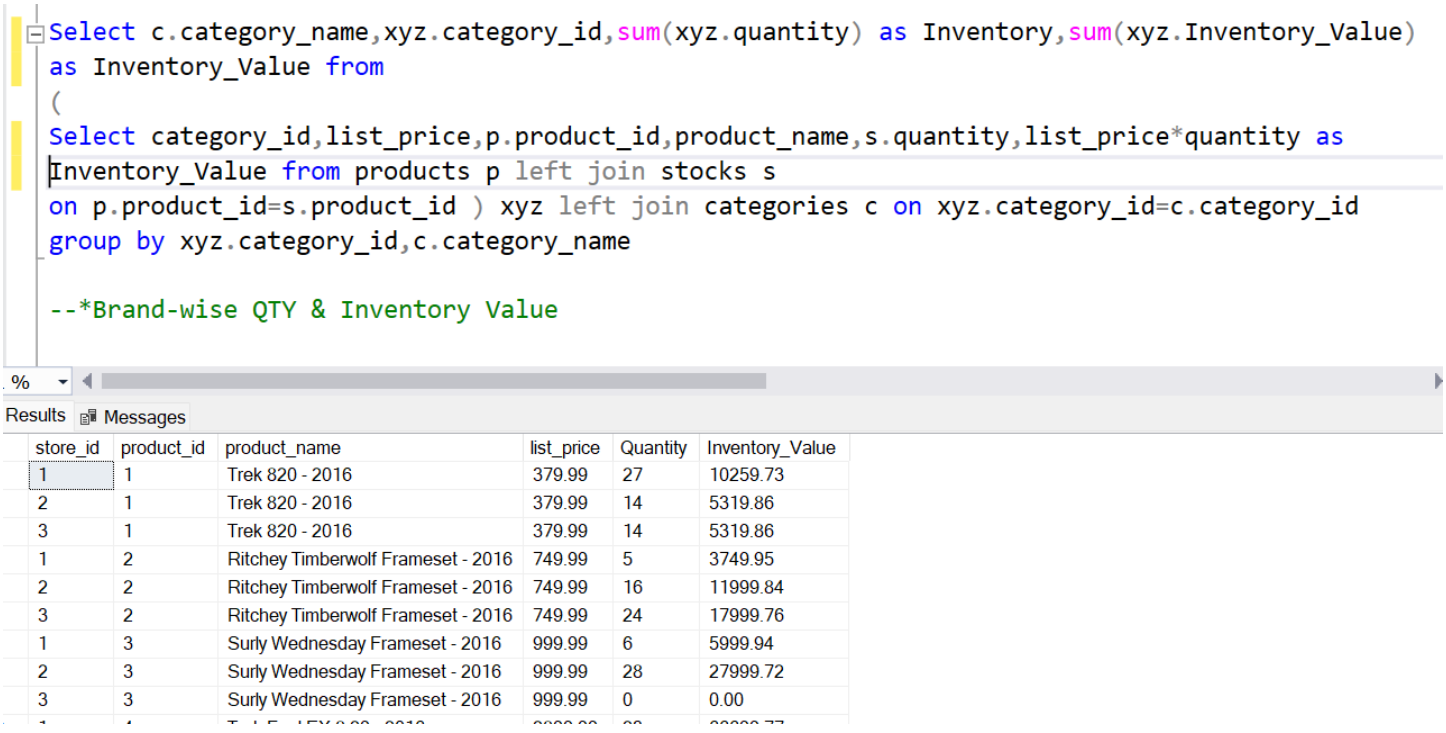
store_id	store_name	city	Quantity	Inventory_Value_Millions
1	Santa Cruz Bikes	Santa Cruz	4532	6.4872421600
2	Baldwin Bikes	Baldwin	4359	6.4731262800
3	Rowlett Bikes	Rowlett	4620	6.8592142100

2.) Query-2

```sql
--*category-wise Inventory QTY & Value of Inventory

Select c.category_name,xyz.category_id,sum(xyz.quantity) as
Inventory,sum(xyz.Inventory_Value) as Inventory_Value from
(
```

```
Select
category_id,list_price,p.product_id,product_name,s.quantity,list_price*quantity
as Inventory_Value from products p left join stocks s
on p.product_id=s.product_id ) xyz left join categories c on
xyz.category_id=c.category_id
group by xyz.category_id,c.category_name
```

screenshot



```
Select c.category_name,xyz.category_id,sum(xyz.quantity) as Inventory,sum(xyz.Inventory_Value)
as Inventory_Value from
(
Select category_id,list_price,p.product_id,product_name,s.quantity,list_price*quantity as
Inventory_Value from products p left join stocks s
on p.product_id=s.product_id ) xyz left join categories c on xyz.category_id=c.category_id
group by xyz.category_id,c.category_name

--*Brand-wise QTY & Inventory Value
```

%

Results | Messages

store_id	product_id	product_name	list_price	Quantity	Inventory_Value
1	1	Trek 820 - 2016	379.99	27	10259.73
2	1	Trek 820 - 2016	379.99	14	5319.86
3	1	Trek 820 - 2016	379.99	14	5319.86
1	2	Ritchey Timberwolf Frameset - 2016	749.99	5	3749.95
2	2	Ritchey Timberwolf Frameset - 2016	749.99	16	11999.84
3	2	Ritchey Timberwolf Frameset - 2016	749.99	24	17999.76
1	3	Surly Wednesday Frameset - 2016	999.99	6	5999.94
2	3	Surly Wednesday Frameset - 2016	999.99	28	27999.72
3	3	Surly Wednesday Frameset - 2016	999.99	0	0.00

3.) Query-3

--*Brand-wise QTY & Inventory Value

```
Select b.brand_name,abc.brand_id,sum(abc.quantity) as
Inventory,sum(abc.Inventory_Value)
as Inventory_value from (
```

```
Select
category_id,p.brand_id,p.product_id,product_name,s.quantity,list_price*quantity
as Inventory_Value from products p left join stocks s
on p.product_id=s.product_id ) abc join brands b on abc.brand_id=b.brand_id
group by b.brand_name,abc.brand_id
order by 2
```

screenshot

```
Select b.brand_name,abc.brand_id,sum(abc.quantity) as Inventory,sum(abc.Inventory_Value)
as Inventory_value from (
Select category_id,p.brand_id,p.product_id,product_name,s.quantity,list_price*quantity
as Inventory_Value from products p left join stocks s
on p.product_id=s.product_id ) abc join brands b on abc.brand_id=b.brand_id
group by b.brand_name,abc.brand_id
order by 2

--*Total_inventory
```

Results Messages

brand_name	brand_id	Inventory	Inventory_value
Electra	1	4998	3761355.97
Haro	2	454	279115.46
Heller	3	108	247463.74
Pure Cycles	4	104	46156.00
Ritchey	5	45	33749.55
Strider	6	136	28798.64
Sun Bicycles	7	1042	516104.58
Surly	8	1105	1483783.90
Trek	9	5519	13423054.81

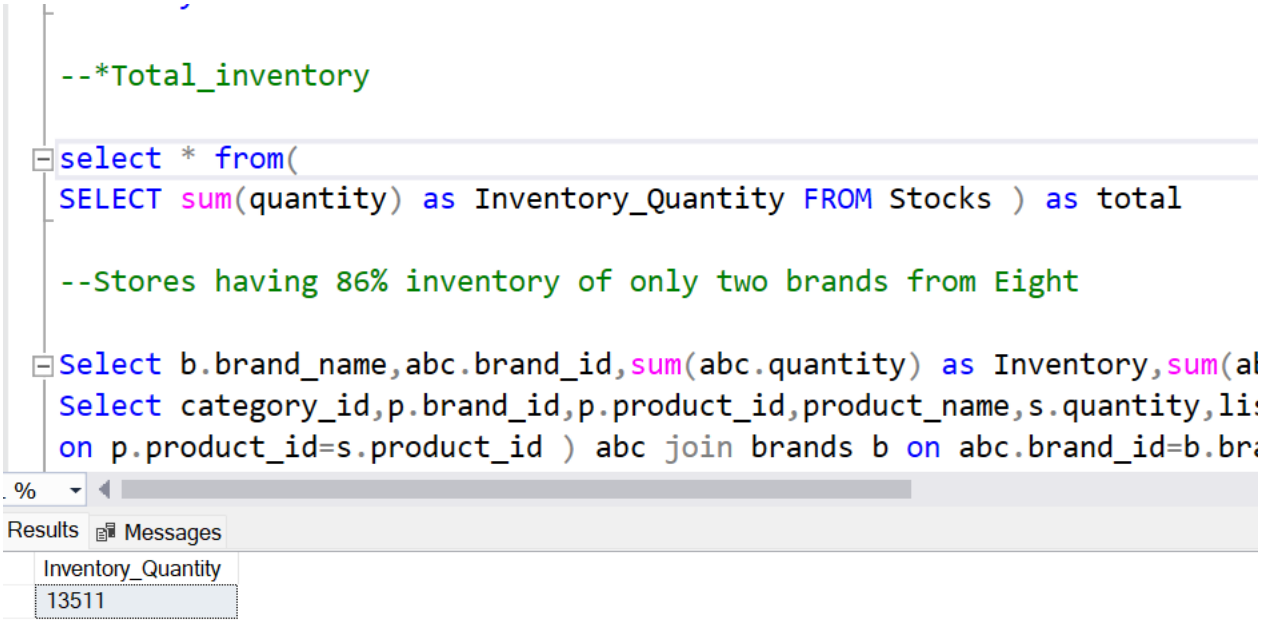
4.) Query-4

--*Total_inventory

```
select * from(
```

```
SELECT sum(quantity) as Inventory_Quantity FROM Stocks ) as total
```

Screenshot

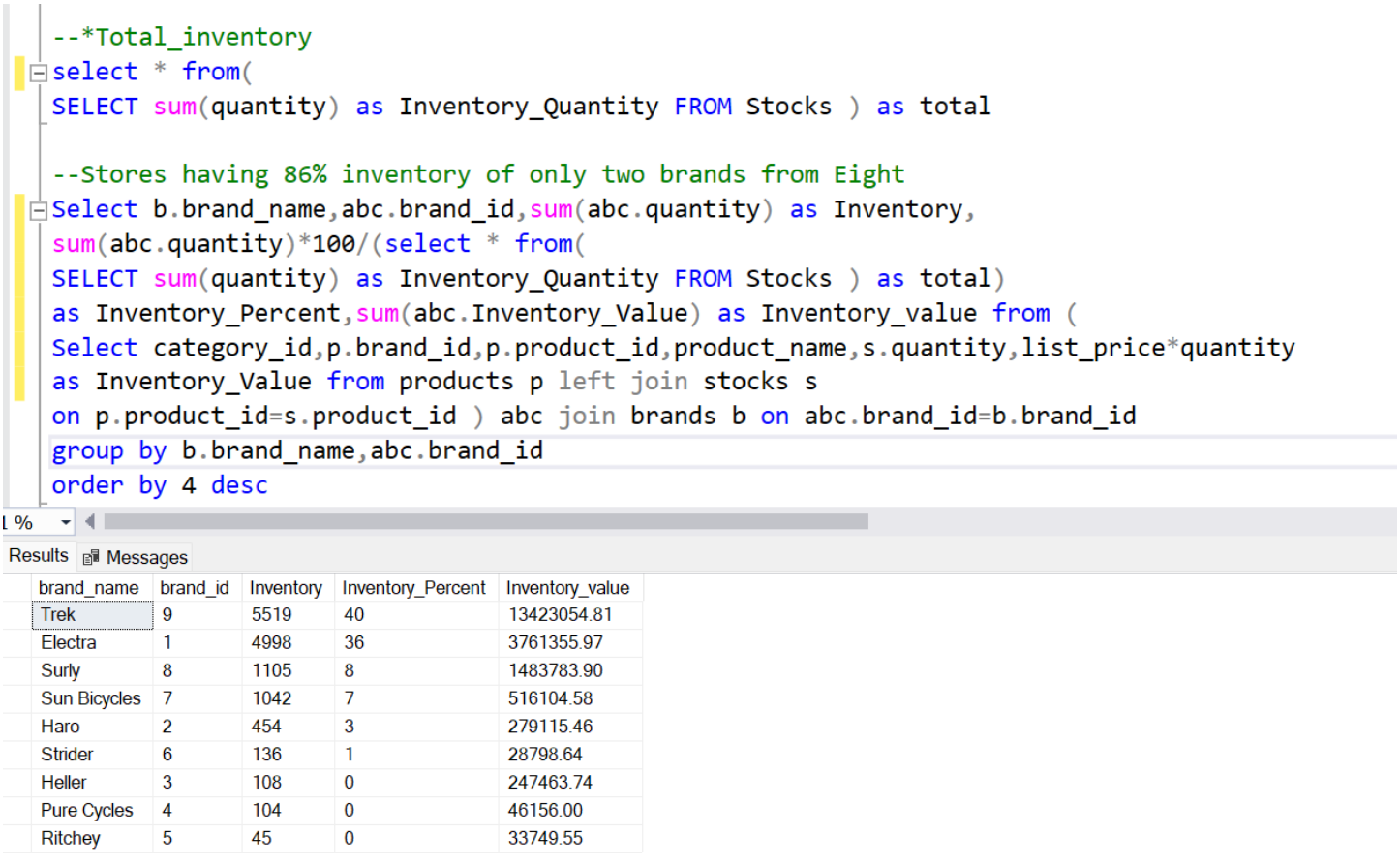


5.) Query-5

```
--Stores having 86% inventory of only two brands from Eight

Select b.brand_name,abc.brand_id,sum(abc.quantity) as Inventory,
sum(abc.quantity)*100/(select * from(
```

```
SELECT sum(quantity) as Inventory_Quantity FROM Stocks ) as total)
as Inventory_Percent,sum(abc.Inventory_Value) as Inventory_value from (
Select
category_id,p.brand_id,p.product_id,product_name,s.quantity,list_price*quantity
as Inventory_Value from products p left join stocks s
on p.product_id=s.product_id ) abc join brands b on abc.brand_id=b.brand_id
group by b.brand_name,abc.brand_id
order by 4 desc
```

screenshot



```
--*Total_inventory
select * from(
SELECT sum(quantity) as Inventory_Quantity FROM Stocks ) as total

--Stores having 86% inventory of only two brands from Eight
Select b.brand_name,abc.brand_id,sum(abc.quantity) as Inventory,
sum(abc.quantity)*100/(select * from(
SELECT sum(quantity) as Inventory_Quantity FROM Stocks ) as total)
as Inventory_Percent,sum(abc.Inventory_Value) as Inventory_value from (
Select category_id,p.brand_id,p.product_id,product_name,s.quantity,list_price*quantity
as Inventory_Value from products p left join stocks s
on p.product_id=s.product_id ) abc join brands b on abc.brand_id=b.brand_id
group by b.brand_name,abc.brand_id
order by 4 desc
```

Results | Messages

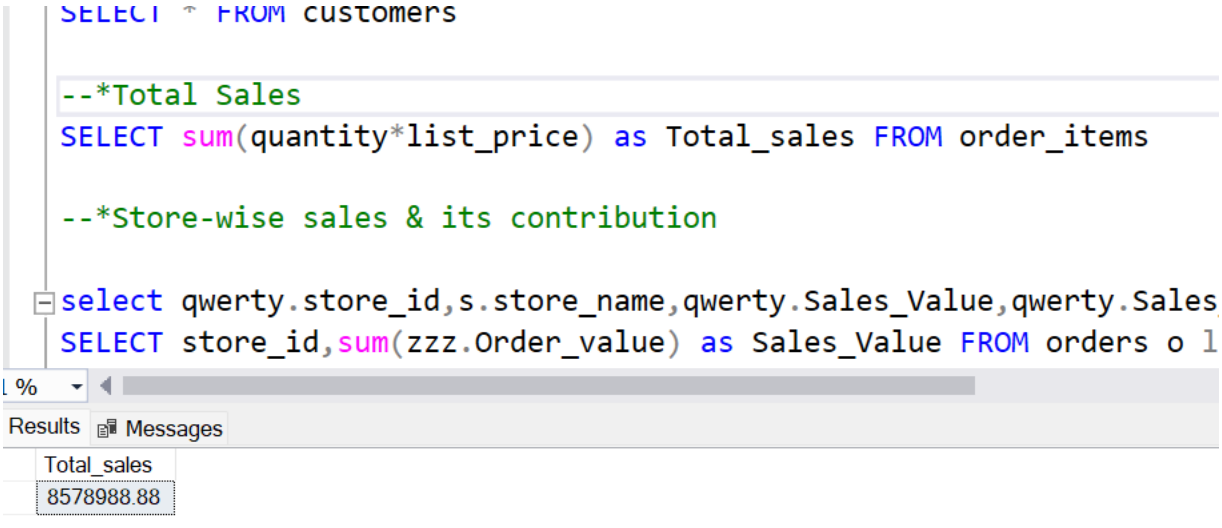
brand_name	brand_id	Inventory	Inventory_Percent	Inventory_value
Trek	9	5519	40	13423054.81
Electra	1	4998	36	3761355.97
Surly	8	1105	8	1483783.90
Sun Bicycles	7	1042	7	516104.58
Haro	2	454	3	279115.46
Strider	6	136	1	28798.64
Heller	3	108	0	247463.74
Pure Cycles	4	104	0	46156.00
Ritchey	5	45	0	33749.55

6.) Query-6

```
--*Total Sales
```

```sql
SELECT sum(quantity*list_price) as Total_sales FROM order_items
```

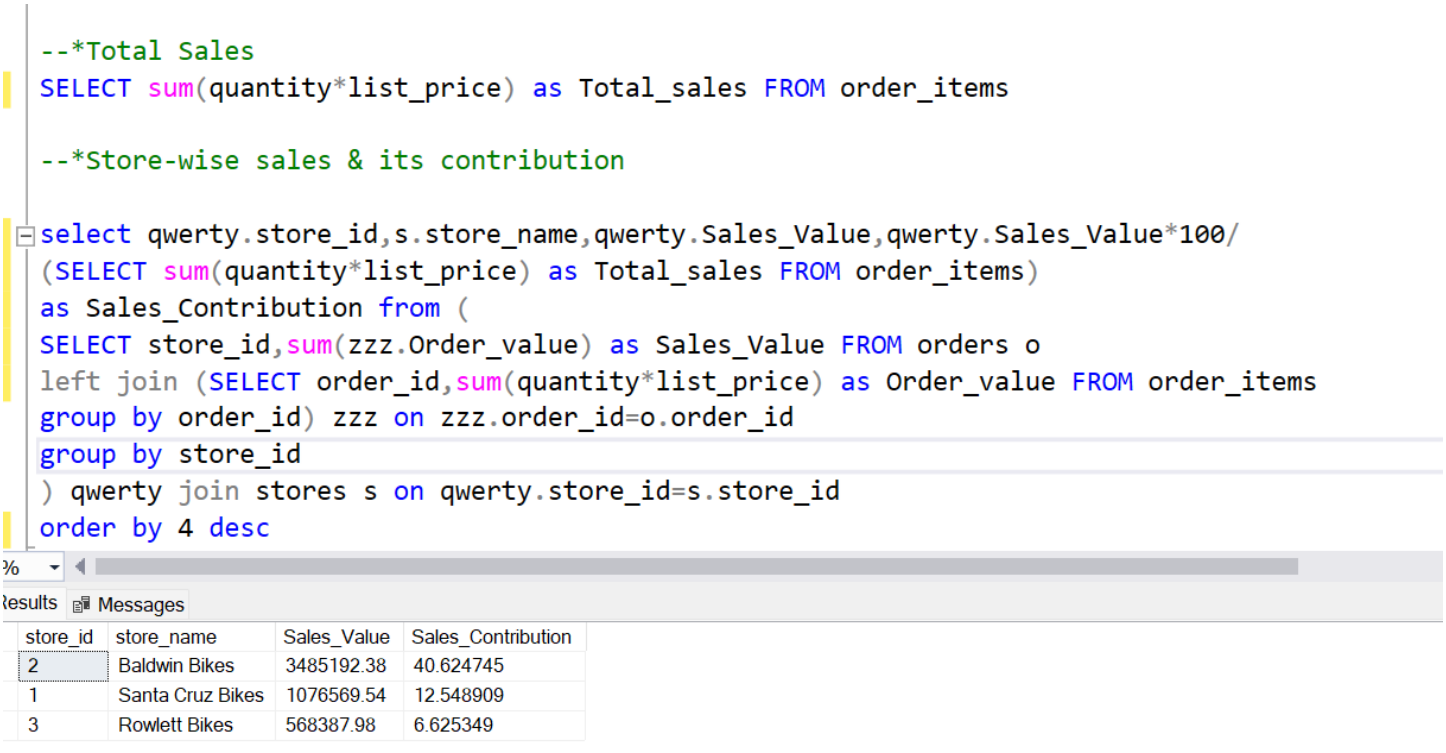
Screenshot



7.) Query-7

```
--*Store-wise sales & its contribution

select qwerty.store_id,s.store_name,qwerty.Sales_Value,qwerty.Sales_Value*100/
(SELECT sum(quantity*list_price) as Total_sales FROM order_items)
as Sales_Contribution from (
SELECT store_id,sum(zzz.Order_value) as Sales_Value FROM orders o
left join (SELECT order_id,sum(quantity*list_price) as Order_value FROM
order_items
group by order_id) zzz on zzz.order_id=o.order_id
group by store_id
) qwerty join stores s on qwerty.store_id=s.store_id
order by 4 desc
```

screenshot



```
--*Total Sales
SELECT sum(quantity*list_price) as Total_sales FROM order_items

--*Store-wise sales & its contribution

select qwerty.store_id,s.store_name,qwerty.Sales_Value,qwerty.Sales_Value*100/
(SELECT sum(quantity*list_price) as Total_sales FROM order_items)
as Sales_Contribution from (
SELECT store_id,sum(zzz.Order_value) as Sales_Value FROM orders o
left join (SELECT order_id,sum(quantity*list_price) as Order_value FROM order_items
group by order_id) zzz on zzz.order_id=o.order_id
group by store_id
) qwerty join stores s on qwerty.store_id=s.store_id
order by 4 desc
```

Results | Messages

store_id	store_name	Sales_Value	Sales_Contribution
2	Baldwin Bikes	3485192.38	40.624745
1	Santa Cruz Bikes	1076569.54	12.548909
3	Rowlett Bikes	568387.98	6.625349

8.) Query-8

```
--**Most product sold

Select pqrs.product_id,pqrs.Sale_value,p.product_name,pqrs.Sale_value*100/8578988
as Sale_percent from (
```

```
SELECT product_id,sum(quantity*list_price) as Sale_value FROM order_items
group by product_id ) pqrs left join products p on p.product_id=pqrs.product_id
order by 4 desc
```

Screenshot

```
--**Most product sold

Select pqrs.product_id,pqrs.Sale_value,p.product_name,pqrs.Sale_value*100/8578988 as
Sale_percent from (
SELECT product_id,sum(quantity*list_price) as Sale_value FROM order_items
group by product_id ) pqrs left join products p on p.product_id=pqrs.product_id
order by 4 desc
```

Results | Messages

product_id	Sale_value	product_name	Sale_percent
7	615998.46	Trek Slash 8 27.5 - 2016	7.180316
9	434998.55	Trek Conduit+ - 2016	5.070511
4	414698.57	Trek Fuel EX 8 29 - 2016	4.833886
11	253829.49	Surly Straggler 650b - 2016	2.958734
56	236499.57	Trek Domane SLR 6 Disc - 2017	2.756730
10	227703.00	Surly Straggler - 2016	2.654194
8	224998.75	Trek Remedy 29 Carbon Frameset - 2016	2.622672
61	204999.59	Trek Powerfly 8 FS Plus - 2017	2.389554
58	194999.61	Trek Madone 9.2 - 2017	2.272990
51	188499.71	Trek Silque SLR 8 Women's - 2017	2.197225
50	179999.70	Trek Silque SLR 7 Women's - 2017	2.098146
43	174899.67	Trek Fuel EX 9.8 27.5 Plus - 2017	2.038698
5	170407.71	Heller Shagamaw Frame - 2016	1.986338

Thank You