SQL ASSIGNMENT 01

Problem Statement:

1.Design the complete database + schema + tables for the diagram shown above using appropriate data type for every column along with any constraints (checks + PK) mentioned in the task description and load the below data into the requisite tables.

- Created a Database Called **BIKE_STORES** and Schema Called **Sales** and **Production**

```
-- CREATED DATABASE BIKE_STORES
CREATE DATABASE BIKE_STORES;

--CREATED SCHEMA SALES
CREATE SCHEMA SALES;

--CREATED SCHEMA PRODUCTION
CREATE SCHEMA PRODUCTION;
```

- Inside **Sales Schema**, I have created **Five Tables**

1) **Customers Table**

```
-- CREATING A CUSTOMERS TABLE

CREATE OR REPLACE TABLE CUSTOMERS
(
CUSTOMER_ID NUMBER(4,0) PRIMARY KEY,
FIRST_NAME VARCHAR(50),
LAST_NAME VARCHAR(50),
PHONE  CHAR(20),
EMAIL VARCHAR(50),
STREET VARCHAR(50),
CITY VARCHAR(50),
STATE CHAR(2),
ZIP_CODE NUMBER(5,0)
);
```

2) Orders Table

```
-- CREATING A ORDERS TABLE

CREATE OR REPLACE TABLE ORDERS
(
ORDER_ID  NUMBER(4,0) PRIMARY KEY,
CUSTOMER_ID NUMBER(4,0),
ORDER_STATUS  NUMBER(1,0),
ORDER_DATE INT,
REQUIRED_DATE INT,
SHIPPED_DATE CHAR(8),
STORE_ID  NUMBER(1,0),
STAFF_ID INT
);
```

3) Order_Items

```
-- CREATING A ORDER_ITEMS TABLE

CREATE OR REPLACE TABLE ORDER_ITEMS
(
ORDER_ID NUMBER(4,0),
ITEM_ID  NUMBER(1,0),
PRODUCT_ID NUMBER(3,0),
QUANTITY NUMBER(1,0),
LIST_PRICE FLOAT,
DISCOUNT FLOAT,
PRIMARY KEY(ORDER_ID,ITEM_ID)

);
```

4) Staffs

```
-- CREATING A STAFFS TABLE

CREATE OR REPLACE TABLE STAFFS
(
STAFF_ID INT PRIMARY KEY,
FIRST_NAME VARCHAR(50),
LAST_NAME VARCHAR(50),
EMAIL VARCHAR(50) NOT NULL,
PHONE VARCHAR(20) NOT NULL,
ACTIVE NUMBER(1,0),
STORE_ID NUMBER(1,0),
MANAGER_ID VARCHAR(10)
);
```

5) Stores

```
--CREATING A STORES TABLE

CREATE OR REPLACE TABLE STORES
(
STORE_ID NUMBER(1,0) PRIMARY KEY,
STORE_NAME VARCHAR(50),
PHONE VARCHAR(20),
EMAIL VARCHAR(20),
STREET VARCHAR(50),
CITY VARCHAR(50),
STATE CHAR(2),
ZIP_CODE NUMBER(5,0)
);
```

- Inside **Production Schema**, I have created **Four Tables**

1) Brands

```
-- CREATING A BRAND TABLE

CREATE OR REPLACE TABLE BRANDS
(
BRAND_ID NUMBER(1,0) PRIMARY KEY,
BRAND_NAME VARCHAR(50)
);
```

2) Categories

```
-- CREATING A CATEGORIES TABLE

CREATE OR REPLACE TABLE CATEGORIES
(
CATEGORY_ID NUMBER(1,0) PRIMARY KEY,
CATEGORY_NAME VARCHAR(50)
);
```

3) Products

```
-- CREATING A PRODUCTS TABLE


CREATE OR REPLACE TABLE PRODUCTS
(
PRODUCT_ID  NUMBER(3,0) PRIMARY KEY,
PRODUCT_NAME VARCHAR(100),
BRAND_ID   NUMBER(1,0),
CATEGORY_ID   NUMBER(1,0),
MODEL_YEAR  CHAR(4),
LIST_PRICE FLOAT
);
```

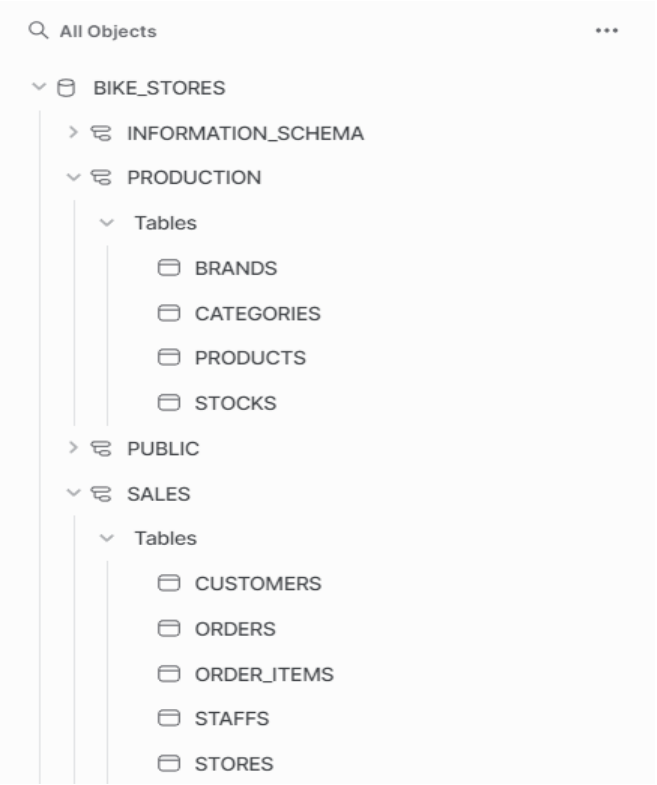
4) Stocks

```
-- CREATING STOCKS TABLE

CREATE OR REPLACE TABLE STOCKS
(
STORE_ID  NUMBER(1,0),
PRODUCT_ID NUMBER(3,0),
QUANTITY NUMBER(2,0),
primary key (STORE_ID,PRODUCT_ID)

);
```

Final Output



- Created a **File Format** For all the **9 tables**.

```
-- CREATE A FILE FORMAT ONCE FOR ALL SO THAT IF WE LOAD DATA FOR 9 TABLES WE DONT HAVE TO SPECIFY AGAIN AND AGAIN.

CREATE OR REPLACE FILE FORMAT BIKE_STORES_FILE_FORMAT
 TYPE = "CSV"
 COMPRESSION = 'NONE',
 FIELD_DELIMITER = ';'
 FIELD_OPTIONALLY_ENCLOSED_BY = 'NONE'
 SKIP_HEADER = 1;
```

2. Once the table has got created, there is a requirement of FOREIGN KEY implementation n coming into picture where one needs to add(ALTER TABLE COMMAND) below foreign key on the table mentioned pointing to another table

- **SALES.STAFFS (STORE_ID) -> SALES.STORES(STORIED)**

Used Alter Command to enforce foreign key on staffs table referencing to stores table

```
-- ADDING FOREIGN KEY CONSTRAINT ON STORE_ID COLUMN OF STAFFS TABLE(CHILD TABLE) WHICH IS REFERENCING STORE_ID COLUMN OF STORES
TABLE(PARENT TABLE)
ALTER TABLE SALES.STAFFS
ADD CONSTRAINT FK_STAFFS_STORES
FOREIGN KEY(STORE_ID) REFERENCES SALES.STORES(STORE_ID);
```

- **SALES.STAFFS (MANAGER_ID) -> SALES.STAFFS (STAFF_ID)**

Adding Foreign Key Constraint On Manager_Id Column Of Staffs Table (Child Table), Which Is Referencing Staff_Id Column Of Staffs Table(Parent Table)

```
ALTER TABLE SALES.STAFFS
ADD CONSTRAINT FK_MANAGER_STAFFS
FOREIGN KEY(MANAGER_ID) REFERENCES SALES.STAFFS(STAFF_ID);
```

- **PRODUCTION.PRODUCTS (CATEGORY_ID) -> PRODUCTION.CATEGORIES (CATEGORY_ID)**

Adding Foreign Key Constraint On Category_Id Column Of Products Table (Child Table), Which Is Referencing Category_Id Column Of Categories Table(Parent Table)

```sql
ALTER TABLE PRODUCTION.PRODUCTS
ADD CONSTRAINT FK_PRODUCTS_CATEGORIES
FOREIGN KEY(CATEGORY_ID) REFERENCES PRODUCTION.CATEGORIES(CATEGORY_ID);
```

- **PRODUCTION.PRODUCTS(BRAND_ID) -> PRODUCTION.BRANDS (BRAND_ID)**

Adding Foreign Key Constraint On Brand_Id Column of Products Table(Child Table),Which Is Referencing Brand_Id Column Of Brands Table(Parent Table)

```sql
ALTER TABLE PRODUCTION.PRODUCTS
ADD CONSTRAINT FK_PRODUCTS_BRANDS
FOREIGN KEY(BRAND_ID) REFERENCES PRODUCTION.BRANDS(BRAND_ID);
```

- **SALES.ORDERS (CUSTOMER_ID) -> SALES.CUSTOMERS (CUSTOMER_ID)**

Adding Foreign Key Constraint On Customer Column Of Orders Table(Child Table), Which Is Referencing Customer_Id Column Of Customers Table(Parent Table)

```sql
ALTER TABLE SALES.ORDERS
ADD CONSTRAINT FK_ORDERS_CUSTOMERS
FOREIGN KEY(CUSTOMER_ID) REFERENCES SALES.CUSTOMERS(CUSTOMER_ID);
```

- **SALES.ORDERS(STORE_ID) -> SALES.STORES (STORE_ID)**

Adding Foreign Key Constraint on Store_Id Column Of Orders Table(Child Table), Which Is Referencing Store_Id Column Of Stores Table(Parent Table)

```sql
ALTER TABLE SALES.ORDERS
ADD CONSTRAINT FK_ORDERS_STORES
FOREIGN KEY(STORE_ID) REFERENCES SALES.STORES(STORE_ID);
```

- **SALES.ORDERS (STAFF_ID) -> SALES.STAFFS (STAFF_ID)**

Adding Foreign Key Constraint On Staff_Id Column Of Orders Table (Child Table), Which Is Referencing Staff_Id Column Of Staffs Table(Parent Table)

```sql
ALTER TABLE SALES.ORDERS
ADD CONSTRAINT FK_ORDERS_STAFFS
FOREIGN KEY(STAFF_ID) REFERENCES SALES.STAFFS(STAFF_ID);
```

- **SALES.ORDER_ITEMS(ORDER_ID) -> SALES.ORDERS (ORDER_ID)**

Adding Foreign Key Constraint on Order_Id Column Of Order_Item Table(Child Table), Which Is Referencing Order_Id Column Of Orders Table(Parent Table)

```sql
ALTER TABLE SALES.ORDER_ITEMS
ADD CONSTRAINT FK_ORDER_ITEMS_ORDERS
FOREIGN KEY(ORDER_ID) REFERENCES SALES.ORDERS(ORDER_ID);
```

- **SALES.ORDER_ITEMS (PRODUCT_ID) -> PRODUCTION.PRODUCTS (PRODUCT_ID)**

Adding Foreign Key Constraint On Product_Id Column Of Order_Item Table (ChildTable) , Which Is Referencing Product_Id Column Of Products Table(Parent Table)

```sql
ALTER TABLE SALES.ORDER_ITEMS
ADD CONSTRAINT FK_ORDER_ITEMS_PRODUCTS
FOREIGN KEY(PRODUCT_ID) REFERENCES PRODUCTION.PRODUCTS(PRODUCT_ID);
```

- **PRODUCTION.STOCKS (STORE_ID) -> SALES.STORES (STORE_ID)**

Adding Foreign Key Constraint on Store_Id Column Of Stocks Table(Child Table), which Is Referencing Store_Id Column Of Stores Table(Parent Table)

```sql
ALTER TABLE PRODUCTION.STOCKS
ADD CONSTRAINT FK_STOCKS_STORES
FOREIGN KEY(STORE_ID) REFERENCES SALES.STORES(STORE_ID);
```

- **PRODUCTION.STOCKS (PRODUCT_ID) -> PRODUCTION.PRODUCTS (PRODUCT_ID)**

Adding Foreign Key Constraint On Product_Id Column Of Stocks Table(Child Table), which Is Referencing Product_Id Column Of Products Table (Parent Table)

```sql
ALTER TABLE PRODUCTION.STOCKS
ADD CONSTRAINT FK_STOCKS_PRODUCTS
FOREIGN KEY(PRODUCT_ID) REFERENCES PRODUCTION.PRODUCTS(PRODUCT_ID);
```

3. Does any of the table has missing or NULL value? If yes which are those and what are their counts?

Yes, **the Customer table**, **Orders Table & Staffs table** has **null values**.

[Customer table as 1267 null values]

```
SELECT COUNT(*) AS NULL_COUNT FROM BIKE_STORES.SALES.CUSTOMERS
WHERE CUSTOMER_ID IS NULL OR FIRST_NAME IS NULL OR LAST_NAME IS NULL OR PHONE IS NULL OR EMAIL IS NULL OR STREET IS NULL  OR
CITY IS NULL OR STATE IS NULL OR ZIP_CODE IS NULL ;
```

OUTPUT

Results Chart

	...	NULL_COUNT
1		1,267

[Orders Table has 170 null values]

```
SELECT COUNT(*) AS NULL_COUNT FROM BIKE_STORES.SALES.ORDERS
WHERE ORDER_ID IS NULL OR CUSTOMER_ID IS NULL OR ORDER_STATUS IS NULL OR  ORDER_DATE IS NULL OR
REQUIRED_DATE IS NULL OR SHIPPED_DATE = 0 OR STORE_ID IS NULL OR  STAFF_ID IS NULL ;
```

OUTPUT

Results Chart

	NULL_COUNT
1	170

[Staffs Table has 1 null values]

```
SELECT COUNT(*) FROM BIKE_STORES.SALES.STAFFS
WHERE STAFF_ID = 0 OR FIRST_NAME IS NULL OR LAST_NAME IS NULL OR EMAIL IS NULL OR PHONE IS NULL
OR ACTIVE IS NULL OR STORE_ID IS NULL OR MANAGER_ID = 0;
```

OUTPUT

ACCOUNTADMIN

Results Chart

	...	NULL_COUNT
1		1

4.Does the datasets has any DUPLICATE (identical rows)? If yes – can you just keep the first record and remove all rest if it's possible without using any JOINS or WINDOW function?

→ No table contains Duplicate Values

5.How many unique tables are present in each schema and under each table how many records are we having? (Write SQL Script for the same – I don't need answer like 3/5/4 etc)?

-- To Show The Unique Table In Sales Schema

```
SHOW TABLES IN SALES;-- TO SHOW THE UNIQUE TABLE IN SALES SCHEMA
```

OUTPUT

	created_on	name	database_name	schema_name	kind	comment	cluster_by	···	rows
1	824 +0000	CUSTOMERS	BIKE_STORES	SALES	TABLE				1,445
2	156 +0000	ORDERS	BIKE_STORES	SALES	TABLE				1,615
3	064 +0000	ORDER_ITEMS	BIKE_STORES	SALES	TABLE				4,722
4	775 +0000	STAFFS	BIKE_STORES	SALES	TABLE				10
5	576 +0000	STORES	BIKE_STORES	SALES	TABLE				3

-- To Show The Unique Table In Production Schema

```
SHOW TABLES IN PRODUCTION;-- TO SHOW THE UNIQUE TABLE IN PRODUCTION SCHEMA
```

OUTPUT

	created_on	name	database_name	schema_name	kind	comment	cluster_by	rows
1	637 +0000	BRANDS	BIKE_STORES	PRODUCTION	TABLE			9
2	329 +0000	CATEGORIES	BIKE_STORES	PRODUCTION	TABLE			7
3	677 +0000	PRODUCTS	BIKE_STORES	PRODUCTION	TABLE			321
4	224 +0000	STOCKS	BIKE_STORES	PRODUCTION	TABLE			939

-- TOTAL NO OF RECORDS IN THE CUSTOMERS TABLE

```
SELECT COUNT(*) FROM BIKE_STORES.SALES.CUSTOMERS;-- TOTAL NO OF RECORDS IN THE CUSTOMERS TABLE
```

OUPUT

	··· COUNT(*)
1	1,445

-- TOTAL NO OF RECORDS IN THE ORDERS TABLE

```
SELECT COUNT(*) FROM BIKE_STORES.SALES.ORDERS; -- TOTAL NO OF RECORDS IN THE ORDERS TABLE
```

OUTPUT

	COUNT(*)
1	1,615

-- TOTAL NO OF RECORDS IN THE ORDER_ITEMS TABLE

```
SELECT COUNT(*) FROM BIKE_STORES.SALES.ORDER_ITEMS; -- TOTAL NO OF RECORDS IN THE ORDER_ITEMS TABLE
```

OUTPUT

	COUNT(*)
1	4,722

-- TOTAL NO OF RECORDS IN THE STAFFS TABLE

```
SELECT COUNT(*) FROM BIKE_STORES.SALES.STAFFS; -- TOTAL NO OF RECORDS IN THE STAFFS TABLE
```

OUTPUT

	COUNT(*)
1	10

-- TOTAL NO OF RECORDS IN THE STORES TABLE

```
SELECT COUNT(*) FROM BIKE_STORES.SALES.STORES; -- TOTAL NO OF RECORDS IN THE STORES TABLE
```

OUTPUT

	COUNT(*)
1	3

-- TOTAL NO OF RECORDS IN THE BRANDS TABLE

```
SELECT COUNT(*) FROM BIKE_STORES.PRODUCTION.BRANDS; -- TOTAL NO OF RECORDS IN THE BRANDS TABLE
```

OUTPUT

	COUNT(*)
1	9

-- TOTAL NO OF RECORDS IN THE CATEGORIES TABLE

```
SELECT COUNT(*) FROM BIKE_STORES.PRODUCTION.CATEGORIES; -- TOTAL NO OF RECORDS IN THE CATEGORIES TABLE
```

OUTPUT

	COUNT(*)
1	7

-- TOTAL NO OF RECORDS IN THE PRODUCTS TABLE

```
SELECT COUNT(*) FROM BIKE_STORES.PRODUCTION.PRODUCTS; -- TOTAL NO OF RECORDS IN THE PRODUCTS TABLE
```

OUTPUT

	··· COUNT(*)
1	321

-- TOTAL NO OF RECORDS IN THE STOCKS TABLE

```
SELECT COUNT(*) FROM BIKE_STORES.PRODUCTION.STOCKS ;-- TOTAL NO OF RECORDS IN THE STOCKS TABLE
```

OUTPUT

	COUNT(*)
1	939

5.How many total serving customer BikeStore has ?

--BikeStore has 1,445 total serving Customer.

```
SELECT COUNT(DISTINCT CUSTOMER_ID) AS total_customers
FROM CUSTOMERS;
```

OUTPUT

	TOTAL_CUSTOMERS
1	1,445

Q6. How many total orders are there ?

--There are total 1,615 Orders.

```
SELECT COUNT( DISTINCT ORDER_ID) AS TOTAL_ORDERS
FROM BIKE_STORES.SALES.ORDERS;
```

OUTPUT

	TOTAL_ORDERS
1	1,615

Q7. Which store has the highest number of sales?

--Baldwin Bikes store has the highest number of Sales.

```
SELECT STORE_ID, STORE_NAME
FROM SALES.STORES
WHERE STORE_ID =
(
SELECT STORE_ID
FROM SALES.ORDERS
GROUP BY 1
ORDER BY COUNT(ORDER_ID) DESC
LIMIT 1);
```

OUTPUT

	STORE_ID	STORE_NAME
1	2	Baldwin Bikes

--Q8 Which month the sales was highest and for which store ?

--In the month of April the sales was highest for Baldwin Bikes.

```sql
SELECT O.STORE_ID AS STORE_ID,
MONTH(O.ORDERS_DATE) AS MON,
S.STORE_NAME,
SUM(OI.LIST_PRICE * OI.QUANTITY - OI.DISCOUNT) AS TOTAL_SALES
FROM SALES.ORDERS O,
SALES.ORDER_ITEMS OI,
SALES.STORES S
WHERE O.ORDER_ID = OI.ORDER_ID AND S.STORE_ID = O.STORE_ID
GROUP BY O.STORE_ID,MON,S.STORE_NAME
ORDER BY  TOTAL_SALES DESC
LIMIT 1;
```

OUTPUT

	STORE_ID	MON	STORE_NAME	TOTAL_SALES
1	2	4	Baldwin Bikes	804,633.13

Q9. How many orders each customer has placed (give me top 10 customers)?

---Top 10 Customer has placed maximum 3 Orders.

```sql
SELECT CUSTOMER_ID,
COUNT(ORDER_ID) AS TOTAL_ORDERS
FROM BIKE_STORES.SALES.ORDERS
GROUP BY CUSTOMER_ID
ORDER BY TOTAL_ORDERS DESC
LIMIT 10;
```

OUTPUT

	CUSTOMER_ID	TOTAL_ORDERS
1	13	3
2	9	3
3	12	3
4	31	3
5	24	3
6	43	3
7	7	3
8	66	3
9	46	3
10	50	3

Q9. Which are the TOP 3 selling products?

--The Top 3 Selling Products are: Trek Slash 8 27.5 – 2016, Trek Conduit+ - 2016, Trek Fuel EX 8 29 - 2016

```
SELECT P.PRODUCT_ID, P.PRODUCT_NAME,
SUM(OI.QUANTITY * OI.LIST_PRICE - OI.DISCOUNT) AS TOTAL_SALES
FROM PRODUCTION.PRODUCTS P,
SALES.ORDER_ITEMS OI
WHERE OI.PRODUCT_ID = P.PRODUCT_ID
GROUP BY 1,2
ORDER BY TOTAL_SALES DESC
LIMIT 3;
```

OUTPUT

	PRODUCT_ID	PRODUCT_NAME ···	TOTAL_SALES
1	7	Trek Slash 8 27.5 - 2016	615,988.44
2	9	Trek Conduit+ - 2016	434,988.03
3	4	Trek Fuel EX 8 29 - 2016	414,687.65

Q10. Which was the first and last order placed by the customer who has placed maximum number of orders?

--The First and the Last Order Placed by Robby Sykes

```
SELECT C.CUSTOMER_ID,C.FIRST_NAME,C.LAST_NAME,MIN(ORDERS_DATE) AS FIRST_ORDER,
MAX(ORDERS_DATE) AS LAST_ORDER,
COUNT(ORDER_ID) AS MAX_ORDER
FROM SALES.ORDERS O,
SALES.CUSTOMERS C
WHERE C.CUSTOMER_ID = O.CUSTOMER_ID
GROUP BY 1,2,3
ORDER BY MAX_ORDER DESC
LIMIT 1;
```

OUTPUT

	CUSTOMER_ID	FIRST_NAME	LAST_NAME	FIRST_ORDER	LAST_ORDER	MAX_ORDER
1	12	Robby	Sykes	2016-03-06	2018-04-23	3

Q11. For every customer , which is the cheapest product and the costliest product which the customer has bought?

```sql
SELECT O.CUSTOMER_ID,
MIN(P.PRODUCT_NAME) AS CHEAPEST_PRODUCT,
MAX(P.PRODUCT_NAME) AS COSTILIEST_PRODUCT
FROM
 SALES.ORDER_ITEMS OI,
 PRODUCTION.PRODUCTS P ,SALES.ORDERS O
WHERE
OI.PRODUCT_ID = P.PRODUCT_ID and O.ORDER_ID= OI.ORDER_ID
GROUP BY
O.CUSTOMER_ID;
```

OUTPUT

	CUSTOMER_ID	CHEAPEST_PRODUCT	COSTILIEST_PRODUCT
1	258	Surly Ice Cream Truck Frameset - 2016	Surly Ice Cream Truck Frameset - 2016
2	1348	Electra Townie Original 7D EQ - 2016	Trek Slash 8 27.5 - 2016
3	923	Surly Straggler - 2016	Trek Conduit+ - 2016
4	583	Electra Girl's Hawaii 1 (20-inch) - 2015/2016	Trek Slash 8 27.5 - 2016
5	1223	Electra Townie Original 7D - 2015/2016	Surly Straggler 650b - 2016
6	403	Electra Townie Original 7D EQ - 2016	Trek Slash 8 27.5 - 2016
7	523	Electra Townie Original 7D EQ - Women's - 2016	Surly Wednesday Frameset - 2016
8	236	Electra Girl's Hawaii 1 (20-inch) - 2015/2016	Trek Slash 8 27.5 - 2016
9	1238	Surly Straggler 650b - 2016	Trek Conduit+ - 2016
10	66	Electra Townie Commute 27D Ladies - 2018	Trek XM700+ Lowstep - 2018

Q12. Which product has orders more than 200 ?

```sql
SELECT P.PRODUCT_NAME,COUNT(DISTINCT OI.ORDER_ID) AS TOTAL_ORDERS
FROM SALES.ORDER_ITEMS OI,
PRODUCTION.PRODUCTS P
WHERE OI.PRODUCT_ID = P.PRODUCT_ID
GROUP BY 1
HAVING TOTAL_ORDERS > 200
ORDER BY TOTAL_ORDERS DESC;
```

OUTPUT

--No Product has more than 200 orders.

	PRODUCT_NAME	TOTAL_ORDERS

Query produced no results

Q13.Add a column TOTAL_PRICE with appropriate data type into the sales. order_items

```
ALTER TABLE SALES.ORDER_ITEMS
ADD COLUMN TOTAL_PRICE FLOAT;
```

OUTPUT

	ORDER_ID	ITEM_ID	··· PRODUCT_ID	QUANTITY	LIST_PRICE	DISCOUNT	TOTAL_PRICE
1	1	1	20	1	599.99	0.2	null
2	1	2	8	2	1,799.99	0.07	null
3	1	3	10	2	1,549	0.05	null
4	1	4	16	2	599.99	0.05	null
5	1	5	4	1	2,899.99	0.2	null
6	2	1	20	1	599.99	0.07	null
7	2	2	16	2	599.99	0.05	null
8	3	1	3	1	999.99	0.05	null

Q14.Calculate TOTAL_PRICE = quantity * list price and update the value for all rows in the sales.order_items table.

```
UPDATE SALES.ORDER_ITEMS
SET TOTAL_PRICE = QUANTITY * LIST_PRICE;
```

OUTPUT

	ORDER_ID	ITEM_ID	PRODUCT_ID	QUANTITY	LIST_PRICE	DISCOUNT	TOTAL_PRICE
1	1	1	20	1	599.99	0.2	599.99
2	1	2	8	2	1,799.99	0.07	3,599.98
3	1	3	10	2	1,549	0.05	3,098
4	1	4	16	2	599.99	0.05	1,199.98
5	1	5	4	1	2,899.99	0.2	2,899.99
6	2	1	20	1	599.99	0.07	599.99
7	2	2	16	2	599.99	0.05	1,199.98
8	3	1	3	1	999.99	0.05	999.99
9	3	2	20	1	599.99	0.05	599.99
10	4	1	2	2	749.99	0.1	1,499.98

Q15.What is the value of the TOTAL_PRICE paid for all the sales. order_items ?

```
SELECT SUM(TOTAL_PRICE) AS TOTAL_PRICE_PAID
FROM SALES.ORDER_ITEMS;
```

OUTPUT

	TOTAL_PRICE_PAID
1	8578988.93