SQL_ASSIGNMENT_1

Problem Statements :

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


```
-- NEW DATABASE CREATED
CREATE DATABASE BIKESTORES;
USE DATABASE BIKESTORES;

-- NEW SCHEMA CREATED

CREATE OR REPLACE SCHEMA PRODUCTION;
CREATE OR REPLACE SCHEMA SALES;

-- TABLE CREATION IN SALES SCHEMA

CREATE OR REPLACE TABLE SALES.STORES
(
 STORE_ID INT  IDENTITY(1,1) ,
 STORE_NAME VARCHAR(25),
 PHONE VARCHAR(25),
 EMAIL VARCHAR(30),
 STREET VARCHAR(30),
 CITY VARCHAR(15),
 STATE VARCHAR(5),
 ZIP_CODE INT ,
 PRIMARY KEY (STORE_ID)
 );

 CREATE OR REPLACE TABLE SALES.STAFFS
 (
  STAFF_ID INT,
  FIRST_NAME VARCHAR(15),
  LAST_NAME VARCHAR(15),
  EMAIL VARCHAR(50) ,
  PHONE VARCHAR(20) ,
  ACTIVE TINYINT,
  STORE_ID INT ,
  MANAGER_ID INT,
  PRIMARY KEY (STAFF_ID)
  );
```

```sql
CREATE OR REPLACE TABLE SALES.CUSTOMERS
(
 CUSTOMER_ID INT IDENTITY(1,1),
 FIRST_NAME VARCHAR(15),
 LAST_NAME VARCHAR(15),
 PHONE VARCHAR(20),
 EMAIL VARCHAR(50),
 STREET VARCHAR(50),
 CITY VARCHAR(50),
 STATE CHAR(10),
 ZIP_CODE INT,
 PRIMARY KEY(CUSTOMER_ID)
);

CREATE OR REPLACE TABLE SALES.ORDERS
(
 ORDER_ID INT,
 CUSTOMER_ID INT,
 ORDER_STATUS INT,
 ORDER_DATE VARCHAR(10),
 REQUIRED_DATE VARCHAR(10),
 SHIPPED_DATE VARCHAR(10),
 STORE_ID INT,
 STAFF_ID INT,
 PRIMARY KEY(ORDER_ID)
);

CREATE OR REPLACE TABLE SALES.ORDER_ITEMS
(
 ORDER_ID INT,
 ITEM_ID INT,
 PRODUCT_ID INT,
 QUANTITY INT,
 LIST_PRICE DECIMAL(10,2),
 DISCOUNT DECIMAL(4,2),
 PRIMARY KEY(ORDER_ID,ITEM_ID)
);
```

```sql
-- TABLE CREATION IN PRODUCTION SCHEMA

CREATE OR REPLACE TABLE PRODUCTION.CATEGORIES
(
 CATEGORY_ID INT,
 CATEGORY_NAME VARCHAR(50),
 PRIMARY KEY(CATEGORY_ID)
);

CREATE OR REPLACE TABLE PRODUCTION.BRANDS
(
 BRAND_ID INT,
 BRAND_NAME VARCHAR(50),
 PRIMARY KEY(BRAND_ID)
);

CREATE OR REPLACE TABLE PRODUCTION.PRODUCTS
(
 PRODUCT_ID INT,
 PRODUCT_NAME VARCHAR(100),
 BRAND_ID INT,
 CATEGORY_ID INT,
 MODEL_YEAR INT,
 LIST_PRICE DECIMAL(10,2),
 PRIMARY KEY(PRODUCT_ID)BIKESTORESBIKESTORES.PRODUCTIONBIKESTORES.SALES
);

CREATE OR REPLACE TABLE PRODUCTION.STOCKS
(
 STORE_ID INT,
 PRODUCT_ID INT,
 QUANTITY INT,
 PRIMARY KEY(STORE_ID, PRODUCT_ID)
);
```
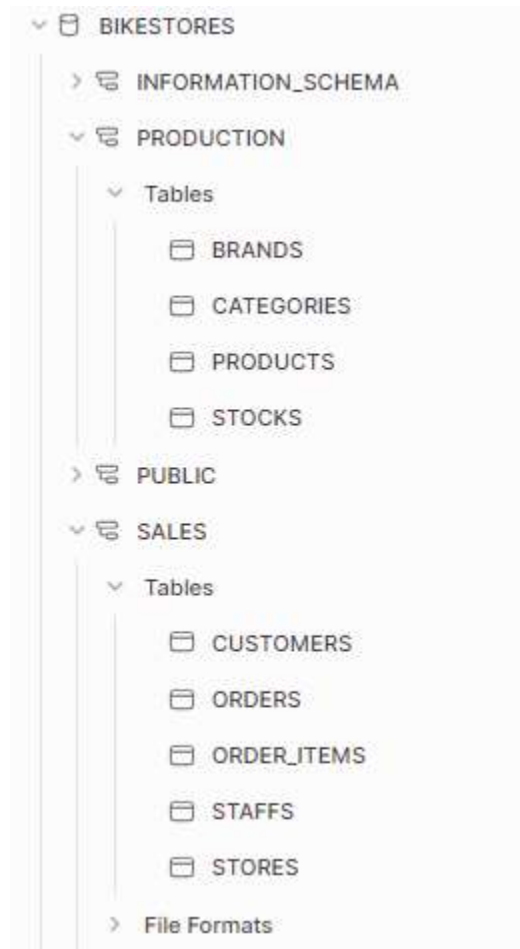
FINAL OUTPUT IS:

- ∨ 🗄 BIKESTORES
  - › 🗃 INFORMATION_SCHEMA
  - ∨ 🗃 PRODUCTION
    - ∨ Tables
      - 🗂 BRANDS
      - 🗂 CATEGORIES
      - 🗂 PRODUCTS
      - 🗂 STOCKS
  - › 🗃 PUBLIC
  - ∨ 🗃 SALES
    - ∨ Tables
      - 🗂 CUSTOMERS
      - 🗂 ORDERS
      - 🗂 ORDER_ITEMS
      - 🗂 STAFFS
      - 🗂 STORES
    - › File Formats

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

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

ALTER TABLE SALES.STAFFS
ADD FOREIGN KEY(STORE_ID) REFERENCES SALES.STORES(STORE_ID);

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

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

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

ALTER TABLE PRODUCTION.PRODUCTS
ADD FOREIGN KEY(CATEGORY_ID) REFERENCES PRODUCTION.CATEGORIES(CATEGORY_ID);

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

ALTER TABLE PRODUCTION.PRODUCTS
ADD FOREIGN KEY(BRAND_ID) REFERENCES PRODUCTION.BRANDS(BRAND_ID);

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

ALTER TABLE SALES.ORDERS
ADD FOREIGN KEY(CUSTOMER_ID) REFERENCES SALES.CUSTOMERS(CUSTOMER_ID);

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

ALTER TABLE SALES.ORDERS
ADD FOREIGN KEY(STORE_ID) REFERENCES SALES.STORES(STORE_ID);

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

ALTER TABLE SALES.ORDERS
ADD FOREIGN KEY(STAFF_ID) REFERENCES SALES.STAFFS(STAFF_ID);

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

ALTER TABLE SALES.ORDER_ITEMS
ADD FOREIGN KEY(ORDER_ID) REFERENCES SALES.ORDERS(ORDER_ID);

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

ALTER TABLE SALES.ORDER_ITEMS
ADD FOREIGN KEY(PRODUCT_ID) REFERENCES PRODUCTION.PRODUCTS(PRODUCT_ID);

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

ALTER TABLE PRODUCTION.STOCKS
ADD FOREIGN KEY(STORE_ID) REFERENCES SALES.STORES(STORE_ID);

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

ALTER TABLE PRODUCTION.STOCKS
ADD 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 ?**

Sales.Customer - Column(PHONE) is having NULL values.

```
181     -- Sales.Customer - Column(PHONE) is having NULL values.
182   | SELECT COUNT(*) AS TOT_NULL_VALUES  FROM SALES.CUSTOMERS WHERE PHONE IS NULL;
183
```

> **Results**   **Chart**

	TOT_NULL_VALUES
1	1,267

Sales.Orders - Column(SHIPPED_DATE) is having NULL values.

```
184     -- Sales.Orders - Column(SHIPPED_DATE) is having NULL values.
185   | SELECT COUNT(*) AS TOT_NULL_VALUES from SALES.ORDERS WHERE SHIPPED_DATE is NULL;
186
187
```

> **Results**   **Chart**

	TOT_NULL_VALUES
1	170

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

SELECT COUNT(*) AS TOT_ROWS FROM SALES.CUSTOMERS; -- **1445**
   SELECT COUNT(DISTINCT FIRST_NAME, LAST_NAME, PHONE, EMAIL, STREET, CITY, STATE, ZIP_CODE) AS TOT_DISTINCT_ROWS
   FROM SALES.CUSTOMERS; --**1445**

   SELECT COUNT(*) AS TOT_ROWS FROM SALES.ORDERS**; -- 1615**
   SELECT COUNT(DISTINCT ORDER_ID, CUSTOMER_ID, ORDER_STATUS, ORDER_DATE, REQUIRED_DATE, SHIPPED_DATE, STORE_ID,STAFF_ID) AS TOT_DISTINCT_ROWS
   FROM SALES.ORDERS; -- **1615**

   SELECT COUNT(*) AS TOT_ROWS FROM SALES.ORDER_ITEMS; -- **4722**
   SELECT COUNT(DISTINCT ORDER_ID, ITEM_ID, PRODUCT_ID, QUANTITY, LIST_PRICE,DISCOUNT) AS TOT_DISTINCT_ROWS
   FROM SALES.ORDER_ITEMS; -- **4722**

   SELECT COUNT(*) AS TOT_ROWS FROM SALES.STAFFS; -- **10**
   SELECT COUNT(DISTINCT STAFF_ID, FIRST_NAME, LAST_NAME, EMAIL, PHONE, ACTIVE, STORE_ID, MANAGER_ID) AS TOT_DISTINCT_ROWS
   FROM SALES.STAFFS; -- **10**

   SELECT COUNT(*) AS TOT_ROWS FROM SALES.STORES; -- **3**
   SELECT COUNT(DISTINCT STORE_NAME, PHONE, EMAIL, STREET, CITY, STATE, ZIP_CODE) AS TOT_DISTINCT_ROWS
   FROM SALES.STORES; -- **3**

   SELECT COUNT(*) AS TOT_ROWS FROM PRODUCTION.BRANDS; -- **9**
   SELECT COUNT(DISTINCT BRAND_ID,BRAND_NAME) AS TOT_DISTINCT_ROWS
   FROM PRODUCTION.BRANDS; -- **9**

   SELECT COUNT(*) AS TOT_ROWS FROM PRODUCTION.CATEGORIES; -- **7**
   SELECT COUNT(DISTINCT CATEGORY_ID,CATEGORY_NAME) AS TOT_DISTINCT_ROWS
   FROM PRODUCTION.CATEGORIES; -- **7**

   SELECT COUNT(*) AS TOT_ROWS FROM PRODUCTION.PRODUCTS; -- **321**
   SELECT COUNT(DISTINCT PRODUCT_ID, PRODUCT_NAME, BRAND_ID, CATEGORY_ID, MODEL_YEAR, LIST_PRICE) AS TOT_DISTINCT_ROWS
   FROM PRODUCTION.PRODUCTS; -- **321**

   SELECT COUNT(*) AS TOT_ROWS FROM PRODUCTION.STOCKS; -- **939**
   SELECT COUNT(DISTINCT STORE_ID, PRODUCT_ID, QUANTITY) AS TOT_DISTINCT_ROWS
   FROM PRODUCTION.STOCKS; -- **939**

Therefore, this dataset has no Duplicate (Identical Rows).

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

```
230    SELECT TABLE_SCHEMA, TABLE_NAME, ROW_COUNT
231    FROM INFORMATION_SCHEMA.TABLES
232    ORDER BY 1 DESC;
233
274
```

↪ Results   ∿ Chart

	TABLE_SCHEMA	TABLE_NAME	ROW_COUNT
1	SALES	ORDERS	1,615
2	SALES	STAFFS	10
3	SALES	STORES	3
4	SALES	CUSTOMERS	1,445
5	SALES	ORDER_ITEMS	4,722
6	PRODUCTION	BRANDS	9
7	PRODUCTION	STOCKS	939
8	PRODUCTION	PRODUCTS	321
9	PRODUCTION	CATEGORIES	7
10	INFORMATION_SCHEMA	TABLES	null
11	INFORMATION_SCHEMA	USAGE_PRIVILEGES	null
12	INFORMATION_SCHEMA	REPLICATION_GROUPS	null
13	INFORMATION_SCHEMA	PROCEDURES	null
14	INFORMATION_SCHEMA	CLASS_INSTANCES	null
15	INFORMATION_SCHEMA	CLASSES	null
16	INFORMATION_SCHEMA	CLASS_INSTANCE_FUNCTIONS	null
17	INFORMATION_SCHEMA	PACKAGES	null
18	INFORMATION_SCHEMA	REPLICATION_DATABASES	null
19	INFORMATION_SCHEMA	FILE_FORMATS	null
20	INFORMATION_SCHEMA	APPLICABLE_ROLES	null

6. How many total serving customer BikeStore has ?

```
237    |    SELECT COUNT(DISTINCT CUSTOMER_ID) AS TOT_SERVING_CUST FROM SALES.ORDERS ;
238
239
```

↳ Results   ∿ Chart

	TOT_SERVING_CUST
1	1,445

7. How many total orders are there ?

```
239
240         -- 7. How many total orders are there ?
241
242    |    SELECT COUNT(DISTINCT ORDER_ID) AS TOT_ORDERS FROM SALES.ORDERS;
243
244
```

↳ Results   ∿ Chart

	TOT_ORDERS
1	1,615

8. Which store has the highest number of sales ?

```
244
245        -- 8. Which store has the highest number of sales ?
246
247        SELECT STORE_ID,
248        COUNT(ORDER_ID) AS Highest_Sales
249        FROM SALES.ORDERS
250        GROUP BY 1
251        ORDER BY 2 DESC
252        LIMIT 1;
253
254
```

↳ Results   ∿ Chart

	STORE_ID	HIGHEST_SALES
1	2	1,093

9. Which month the sales was highest and for which store ?

```
254
255        -- 9. Which month the sales was highest and for which store ?
256
257        SELECT SUBSTR(ORDER_DATE,6,2) AS Month_No,
258        ORDERS.STORE_ID,
259        ROUND(SUM(QUANTITY*LIST_PRICE*(1-DISCOUNT)),2) AS TOT_SALES
260        FROM BIKESTORES.SALES.ORDER_ITEMS, BIKESTORES.SALES.ORDERS
261        GROUP BY 1,2
262        ORDER BY 3 DESC
263        LIMIT 1;
264
```

↳ Results   ∿ Chart

	MONTH_NO	STORE_ID	TOT_SALES
1	03	2	1,022,652,502.16

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

```
265
266        -- 10. How many orders each customer has placed (give me top 10 customers)
267
268        SELECT CUSTOMER_ID,
269        COUNT(DISTINCT ORDER_ID) AS TOT_ORDERS
270        FROM BIKESTORES.SALES.ORDERS
271        GROUP BY 1
272        ORDER BY 2 DESC
273        LIMIT 10;
274
```

↳ **Results** ∿ Chart

	CUSTOMER_ID	TOT_ORDERS
1	32	3
2	8	3
3	7	3
4	31	3
5	66	3
6	9	3
7	50	3
8	5	3
9	43	3
10	24	3

11. Which are the TOP 3 selling product ?

```sql
275
276        -- 11. Which are the TOP 3 selling product ?
277
278        SELECT PRODUCT_ID,
279        ROUND(SUM(QUANTITY*LIST_PRICE*(1-DISCOUNT)),2) AS TOT_SALES
280        FROM BIKESTORES.SALES.ORDER_ITEMS
281        GROUP BY 1
282        ORDER BY 2 DESC
283        LIMIT 3;
284
285
```

↳ Results   ∿ Chart

	PRODUCT_ID	TOT_SALES
1	7	555,558.61
2	9	389248.70
3	4	368,472.73

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

```sql
285
286        -- 12. Which was the first and last order placed by the customer who has placed maximum number of orders ?
287
288        SELECT CUSTOMER_ID,
289        MIN(ORDER_ID) AS First_Order,
290        MAX(ORDER_ID) as Last_Order
291        FROM BIKESTORES.SALES.ORDERS
292        GROUP BY 1
293        ORDER BY COUNT(ORDER_ID) DESC
294        LIMIT 1;
295
```

↳ Results   ∿ Chart

	CUSTOMER_ID	FIRST_ORDER	LAST_ORDER
1	7	104	1,604

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

BIKESTORES.SALES ▼   Settings ▼

```
297
298        -- 13. For every customer , which is the cheapest product and the costliest product which the customer has bought.
299
300            SELECT Cheap.CUSTOMER_ID, Cheapest_Product, Costliest_Product
301            FROM
302                (
303                    SELECT CUSTOMER_ID, PRODUCT_NAME AS Cheapest_Product
304                    FROM
305                        (
306                            SELECT CUSTOMER_ID, PRODUCT_NAME,
307                            DENSE_RANK() OVER (PARTITION BY CUSTOMER_ID ORDER BY OT.LIST_PRICE ASC) AS PRICE_RANK
308                            FROM BIKESTORES.SALES.ORDERS O
309                            INNER JOIN BIKESTORES.SALES.ORDER_ITEMS OT
310                            ON O.ORDER_ID = OT.ORDER_ID
311                            INNER JOIN BIKESTORES.PRODUCTION.PRODUCTS P
312                            ON OT.PRODUCT_ID = P.PRODUCT_ID
313                            ORDER BY 1,3
314                        )
315                    WHERE PRICE_RANK = 1
316                ) Cheap
317
318            INNER JOIN
319
320                (
321                    SELECT CUSTOMER_ID, PRODUCT_NAME AS Costliest_Product
322                    FROM
323                        (
324                            SELECT CUSTOMER_ID, PRODUCT_NAME,
325                            DENSE_RANK() OVER (PARTITION BY CUSTOMER_ID ORDER BY OT.LIST_PRICE DESC) AS PRICE_RANK
326                            FROM BIKESTORES.SALES.ORDERS O
327                            INNER JOIN BIKESTORES.SALES.ORDER_ITEMS OT
328                            ON O.ORDER_ID = OT.ORDER_ID
329                            INNER JOIN BIKESTORES.PRODUCTION.PRODUCTS P
330                            ON OT.PRODUCT_ID = P.PRODUCT_ID
331                            ORDER BY 1,3
332                        )
333                    WHERE PRICE_RANK = 1
334                ) Costly
335            ON Cheap.CUSTOMER_ID = Costly.CUSTOMER_ID;          ;
336
```

↳ Results   ∿ Chart

	CUSTOMER_ID	CHEAPEST_PRODUCT	COSTLIEST_PRODUCT
1	1	"Electra Girl's Hawaii 1 (16-inch) - 2015/2016"	"Trek Domane SL Frameset - 2018"
2	2	"Sun Bicycles Lil Bolt Type-R - 2017"	"Trek Domane SLR 6 Disc - 2017"
3	3	"Trek Precaliber 12 Boy's - 2018"	"Trek Silque SLR 7 Women's - 2017"
4	4	"Strider Strider 20 Sport - 2018"	"Trek Slash 8 27.5 - 2016"

14. Which product has orders more than 200 ?

There is no product who has more than 200 orders

```
341    -- 14. Which product has orders more than 200 ?
342
343    SELECT PRODUCT_ID, COUNT(DISTINCT ORDER_ID) AS TOT_ORDERS
344    FROM BIKESTORES.SALES.ORDER_ITEMS
345    GROUP BY 1
346    HAVING TOT_ORDERS > 200
347    ORDER BY 2 DESC;
348
```

↳ Results   ∕ Chart

PRODUCT_ID	TOT_ORDERS
Query produced no results	

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

```
-- 15. Add a column TOTAL_PRICE with appropriate data type into the sales.order_items

ALTER TABLE BIKESTORES.SALES.ORDER_ITEMS
ADD COLUMN TOTAL_PRICE DECIMAL(11,2);
```

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

```
-- 16. Calculate TOTAL_PRICE = quantity * list price and Update the value for all rows in the sales.order_items table.

UPDATE BIKESTORES.SALES.ORDER_ITEMS
SET TOTAL_PRICE = ROUND(QUANTITY * LIST_PRICE,2);
```

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

```
362        -- 17. What is the value of the TOTAL_PRICE paid for all the sales.order_items ?
363
364   |    SELECT SUM(TOTAL_PRICE) AS TOT_PRICE_PAID FROM BIKESTORES.SALES.ORDER_ITEMS;
365
```

↳ Results   ∕ Chart

	TOT_PRICE_PAID
1	8578988.88

************** THANK YOU **************