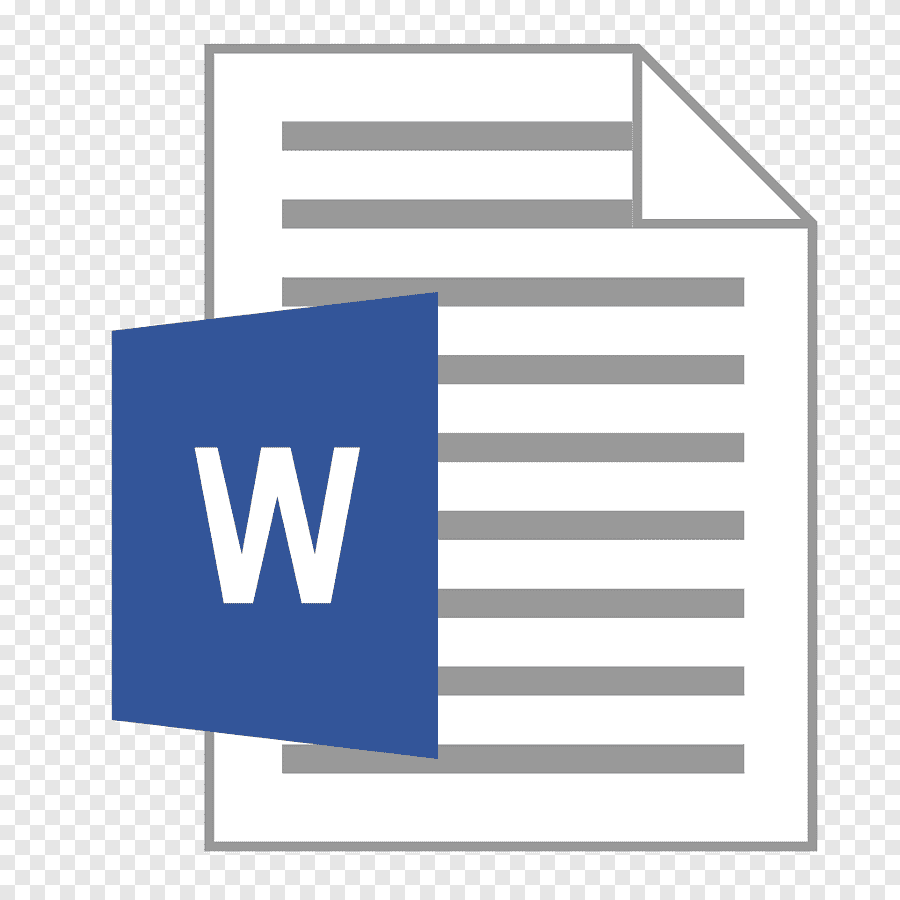
**SQL\_ASSIGNMENT\_1**

**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.**

[](https://github.com/shivampanwar1507/MASTER-DATA-ANALYTICS-USING-CLOUD-TECHNOLOGIES-ML/blob/main/Snowflake/Assignments/SQL_ASSIGNMENT_1_2023_08_29/bike_store_insert_data.txt)

**-- 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)

);

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)

);

**-- 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:**

**A screenshot of a computer

Description automatically generated**

1. **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);

1. **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.**

SELECT COUNT(\*) AS TOT\_NULL\_VALUES FROM SALES.CUSTOMERS WHERE PHONE IS NULL;

A screenshot of a phone

Description automatically generated

**Sales.Orders - Column(SHIPPED\_DATE) is having NULL values.**

SELECT COUNT(\*) AS TOT\_NULL\_VALUES from SALES.ORDERS WHERE SHIPPED\_DATE is NULL;

A screenshot of a computer

Description automatically generated

1. **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**

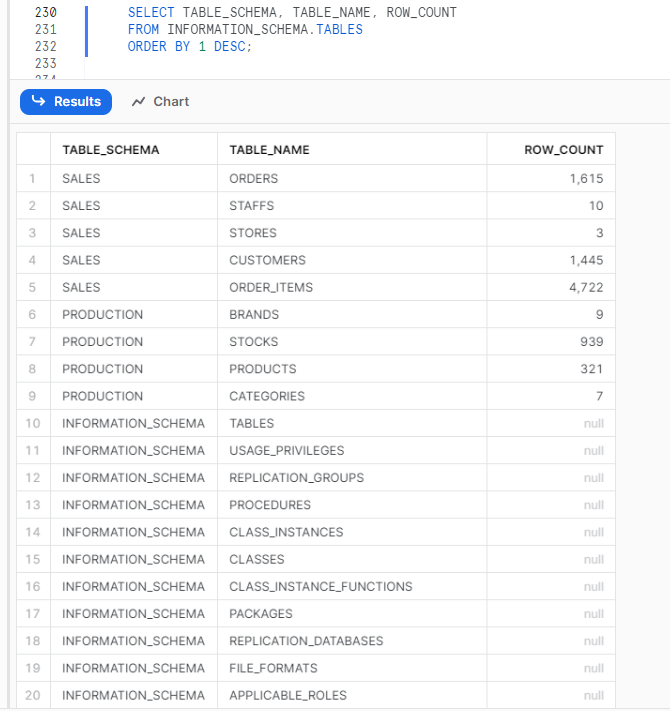
**Therefore, this dataset has no Duplicate (Identical Rows).**

1. **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)**

SELECT TABLE\_SCHEMA, TABLE\_NAME, ROW\_COUNT

FROM INFORMATION\_SCHEMA.TABLES

ORDER BY 1 DESC;



1. **How many total serving customer BikeStore has ?**

SELECT COUNT(DISTINCT CUSTOMER\_ID) AS TOT\_SERVING\_CUST FROM SALES.ORDERS ;

A screenshot of a computer

Description automatically generated

1. **How many total orders are there ?**

SELECT COUNT(DISTINCT ORDER\_ID) AS TOT\_ORDERS FROM SALES.ORDERS;

A screenshot of a computer

Description automatically generated

1. **Which store has the highest number of sales ?**

SELECT STORE\_ID, COUNT(ORDER\_ID) AS Highest\_Sales

FROM SALES.ORDERS

GROUP BY 1

ORDER BY 2 DESC

LIMIT 1;

A screenshot of a computer

Description automatically generated

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

SELECT SUBSTR(ORDER\_DATE,6,2) AS Month\_No,

ORDERS.STORE\_ID,

ROUND(SUM(QUANTITY\*LIST\_PRICE\*(1-DISCOUNT)),2) AS TOT\_SALES

FROM BIKESTORES.SALES.ORDER\_ITEMS, BIKESTORES.SALES.ORDERS

GROUP BY 1,2

ORDER BY 3 DESC

LIMIT 1;

A screenshot of a computer

Description automatically generated

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

SELECT CUSTOMER\_ID,

COUNT(DISTINCT ORDER\_ID) AS TOT\_ORDERS

FROM BIKESTORES.SALES.ORDERS

GROUP BY 1

ORDER BY 2 DESC

LIMIT 10;

A screenshot of a computer

Description automatically generated

1. **Which are the TOP 3 selling product ?**

SELECT PRODUCT\_ID,

ROUND(SUM(QUANTITY\*LIST\_PRICE\*(1-DISCOUNT)),2) AS TOT\_SALES

FROM BIKESTORES.SALES.ORDER\_ITEMS

GROUP BY 1

ORDER BY 2 DESC

LIMIT 3;

A screenshot of a computer

Description automatically generated

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

SELECT CUSTOMER\_ID,

MIN(ORDER\_ID) AS First\_Order,

MAX(ORDER\_ID) as Last\_Order

FROM BIKESTORES.SALES.ORDERS

GROUP BY 1

ORDER BY COUNT(ORDER\_ID) DESC

LIMIT 1;

A screenshot of a computer

Description automatically generated

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

SELECT Cheap.CUSTOMER\_ID, Cheapest\_Product, Costliest\_Product

FROM

(

SELECT CUSTOMER\_ID, PRODUCT\_NAME AS Cheapest\_Product

FROM

(

SELECT CUSTOMER\_ID, PRODUCT\_NAME,

DENSE\_RANK() OVER (PARTITION BY CUSTOMER\_ID ORDER BY OT.LIST\_PRICE ASC) AS PRICE\_RANK

FROM BIKESTORES.SALES.ORDERS O

INNER JOIN BIKESTORES.SALES.ORDER\_ITEMS OT

ON O.ORDER\_ID = OT.ORDER\_ID

INNER JOIN BIKESTORES.PRODUCTION.PRODUCTS P

ON OT.PRODUCT\_ID = P.PRODUCT\_ID

ORDER BY 1,3

)

WHERE PRICE\_RANK = 1

) Cheap

INNER JOIN

(

SELECT CUSTOMER\_ID, PRODUCT\_NAME AS Costliest\_Product

FROM

(

SELECT CUSTOMER\_ID, PRODUCT\_NAME,

DENSE\_RANK() OVER (PARTITION BY CUSTOMER\_ID ORDER BY OT.LIST\_PRICE DESC) AS PRICE\_RANK

FROM BIKESTORES.SALES.ORDERS O

INNER JOIN BIKESTORES.SALES.ORDER\_ITEMS OT

ON O.ORDER\_ID = OT.ORDER\_ID

INNER JOIN BIKESTORES.PRODUCTION.PRODUCTS P

ON OT.PRODUCT\_ID = P.PRODUCT\_ID

ORDER BY 1,3

)

WHERE PRICE\_RANK = 1

) Costly

ON Cheap.CUSTOMER\_ID = Costly.CUSTOMER\_ID;

A screenshot of a computer

Description automatically generated

1. **Which product has orders more than 200 ?**

SELECT PRODUCT\_ID, COUNT(DISTINCT ORDER\_ID) AS TOT\_ORDERS

FROM BIKESTORES.SALES.ORDER\_ITEMS

GROUP BY 1

HAVING TOT\_ORDERS > 200

ORDER BY 2 DESC;

There is no product who has more than 200 orders

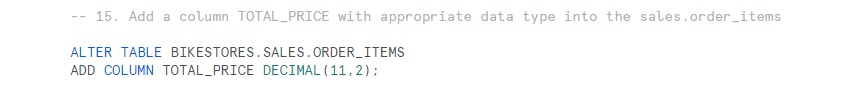
A screenshot of a computer

Description automatically generated

1. **Add a column TOTAL\_PRICE with appropriate data type into the sales.order\_items.**

ALTER TABLE BIKESTORES.SALES.ORDER\_ITEMS

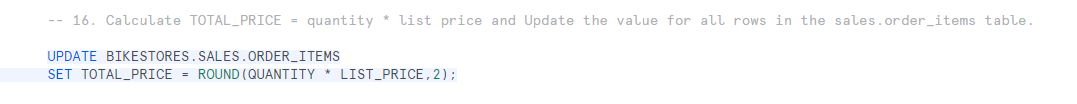
ADD COLUMN TOTAL\_PRICE DECIMAL(11,2);



1. **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);



1. **What is the value of the TOTAL\_PRICE paid for all the sales.order\_items ?**

SELECT SUM(TOTAL\_PRICE) AS TOT\_PRICE\_PAID FROM BIKESTORES.SALES.ORDER\_ITEMS;

A screen shot of a computer

Description automatically generated

**\*\*\*\*\*\*\*\*\*\*\*\*\*\* THANK YOU \*\*\*\*\*\*\*\*\*\*\*\*\*\*\***