1.Design the complete database + schema + tables

```
create DATABASE Bikestores;
USE Bikestores;
CREATE OR REPLACE SCHEMA Sales;
CREATE OR REPLACE SCHEMA Production;
```

--table creation in sales--

1.
```
CREATE OR REPLACE TABLE  Sales.customers
(customer_id int AUTOINCREMENT,
 first_name VARCHAR(50),
 last_name VARCHAR(50),
 phone VARCHAR(15),
 email VARCHAR(60),
 street VARCHAR(100),
 city VARCHAR(50),
 state VARCHAR(50),
 zip_code VARCHAR(50),
 PRIMARY KEY(customer_id)
);
```

2
```
CREATE OR REPLACE TABLE Sales.orders
 (
  order_id VARCHAR(50) NOT NULL PRIMARY KEY,
  customer_id VARCHAR(15),
  order_status VARCHAR(50),
  order_date DATE,
  required_date DATE,
  shipped_date DATE,
  store_id INT,
  staff_id INT
);
```

3.
```
CREATE TABLE if not exists Sales.staffs
(
 staff_id INT PRIMARY KEY,
 first_name VARCHAR(50),
```

```
last_name VARCHAR(50),
phone VARCHAR(15),
email VARCHAR(60),
active VARCHAR(100),
store_id INT,
manager_id INT
);
```

4.
```
CREATE TABLE if not exists Sales.stores
(
store_id int AUTOINCREMENT PRIMARY KEY,
store_name VARCHAR(50),
phone VARCHAR(15),
email VARCHAR(60),
street VARCHAR(100),
city VARCHAR(50),
state VARCHAR(50),
zip_code INT
);
```

5.
```
CREATE TABLE if not exists Sales.order_items
(
order_id INT,
item_id INT,
product_id INT,
quantity VARCHAR(50),
list_price VARCHAR(50),
discount VARCHAR(50),
primary key(order_id, item_id)
);
```

```
--CREATING PRODUCTION TABLE  --
```

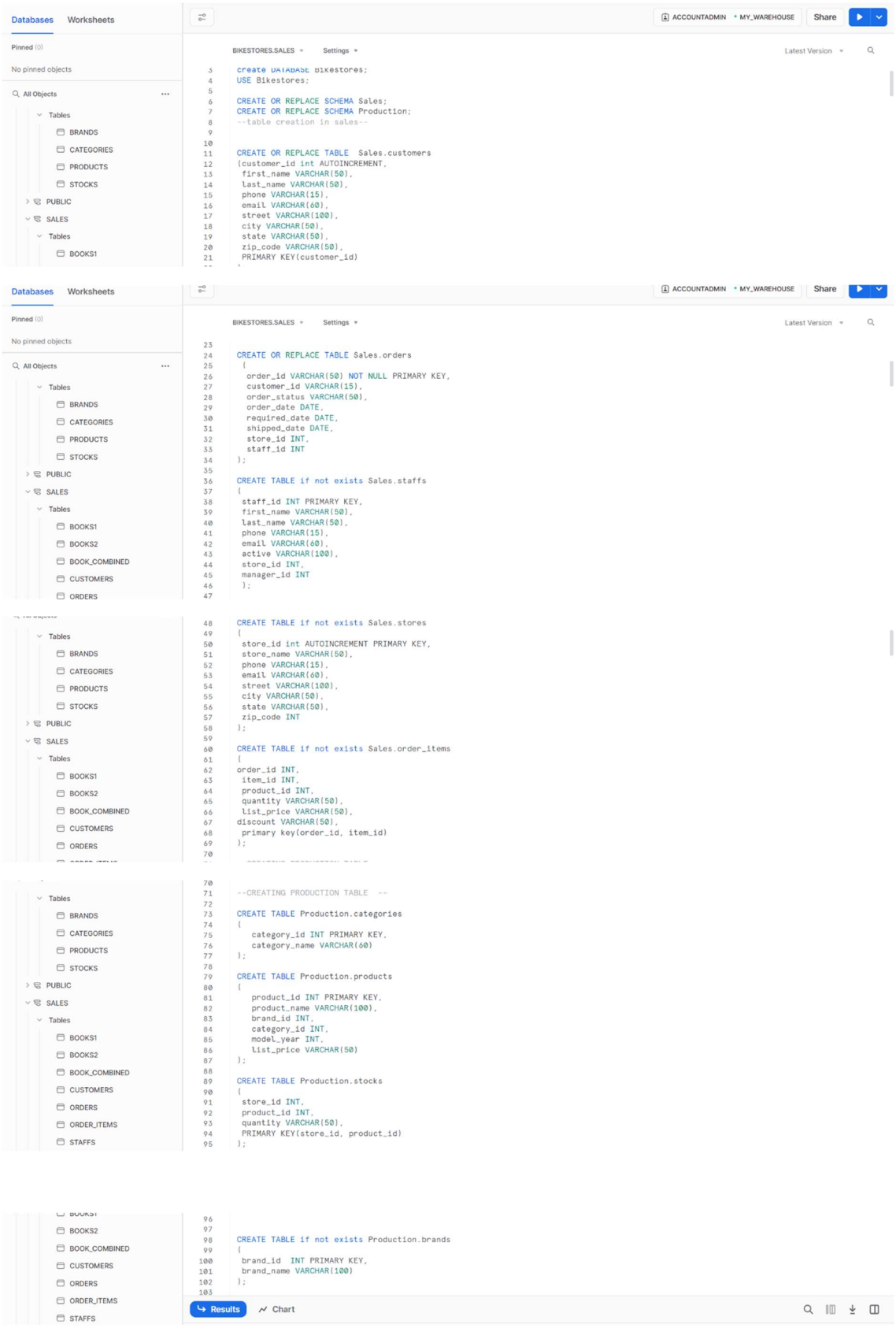
6.
```
CREATE TABLE Production.categories
```

```
(
  category_id INT PRIMARY KEY,
  category_name VARCHAR(60)
);
```

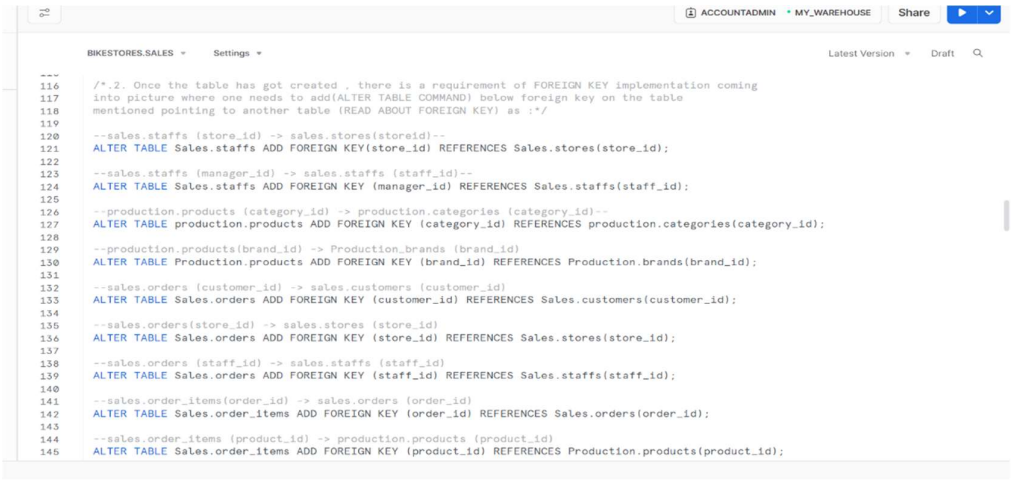
7.
```
CREATE TABLE Production.products
(
  product_id INT PRIMARY KEY,
  product_name VARCHAR(100),
  brand_id INT,
  category_id INT,
  model_year INT,
  list_price VARCHAR(50)
);
```

8.
```
CREATE TABLE Production.stocks
(
 store_id INT,
 product_id INT,
 quantity VARCHAR(50),
 PRIMARY KEY(store_id, product_id)
);
```

9.
```
CREATE TABLE if not exists Production.brands
(
 brand_id  INT PRIMARY KEY,
 brand_name VARCHAR(100)
);
```

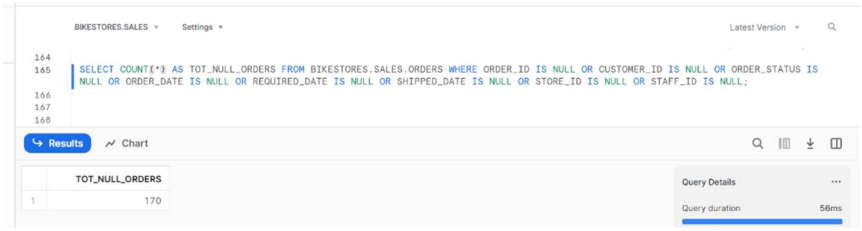


2. Once the table has got created , there is a requirement of FOREIGN KEY implementation coming into picture where one needs to add(ALTER TABLE COMMAND) below foreign key on the table mentioned pointing to another table (READ ABOUT FOREIGN KEY) as :

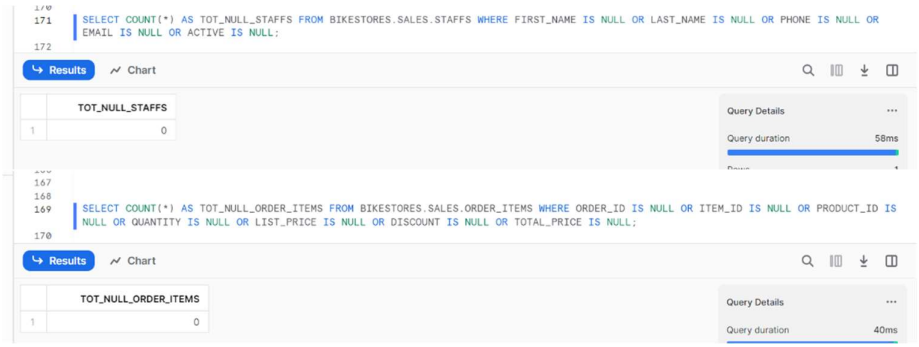


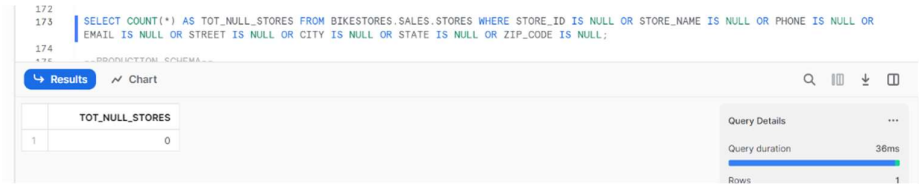
3. Does any of the table has missing or NULL value ? If yes which are those and what are their counts?
SELECT COUNT(*) FROM SALES.CUSTOMERS WHERE PHONE IS NULL;--1267—(in snowflake o/p is 0 but in mysql is giving 1267 o/p)

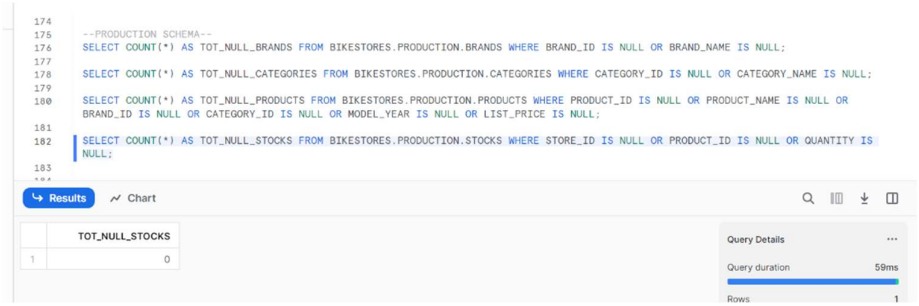
SELECT COUNT(*) FROM SALES.ORDERS WHERE SHIPPED_DATE IS NULL;--170—



No null values present further..







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

THERE IS NO DUPLICATE RECORS.

I Used:
SELECT COUNT(DISTINCT COLUMNS_NAMES) FROM TABLE;
And then compare above command with:
SELECT COUNT(COLUMNS_NAMES) FROM TABLE;

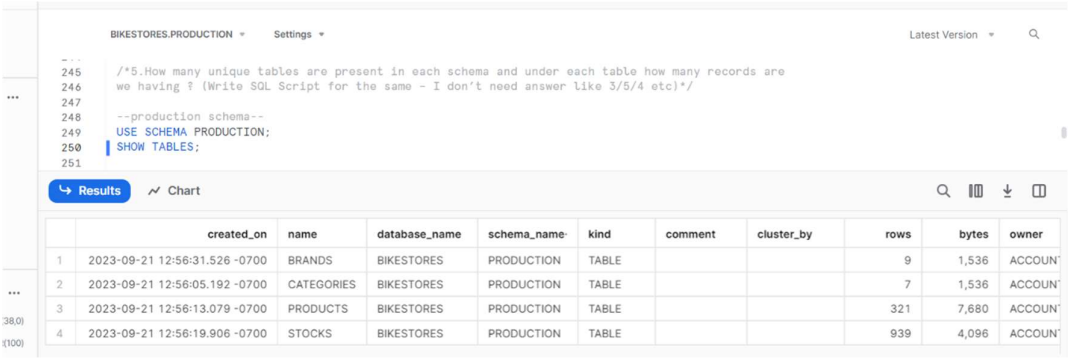
If both the counts are same then we will say there are no duplicate records present.

```
BIKESTORES.SALES   Settings

--CATEGORIES--
SELECT COUNT(CATEGORY_ID, CATEGORY_NAME) FROM BIKESTORES.PRODUCTION.CATEGORIES; --7--
SELECT COUNT(DISTINCT CATEGORY_ID, CATEGORY_NAME) FROM BIKESTORES.PRODUCTION.CATEGORIES; --7--

--PRODUCTS--
SELECT COUNT(PRODUCT_ID, PRODUCT_NAME, BRAND_ID, CATEGORY_ID, MODEL_YEAR, LIST_PRICE)
FROM BIKESTORES.PRODUCTION.PRODUCTS; --321--
SELECT COUNT(DISTINCT PRODUCT_ID, PRODUCT_NAME, BRAND_ID, CATEGORY_ID, MODEL_YEAR, LIST_PRICE)
FROM BIKESTORES.PRODUCTION.PRODUCTS; --321--

--STOCKS--
SELECT COUNT(STORE_ID, PRODUCT_ID, QUANTITY) FROM BIKESTORES.PRODUCTION.STOCKS; --939--
SELECT COUNT(DISTINCT STORE_ID, PRODUCT_ID, QUANTITY) FROM BIKESTORES.PRODUCTION.STOCKS; --939--
```

5.How many unique tables are present in each schema and under each table how many records are we having ?
For Schema Production:

USE SCHEMA PRODUCTION;

SHOW TABLES;

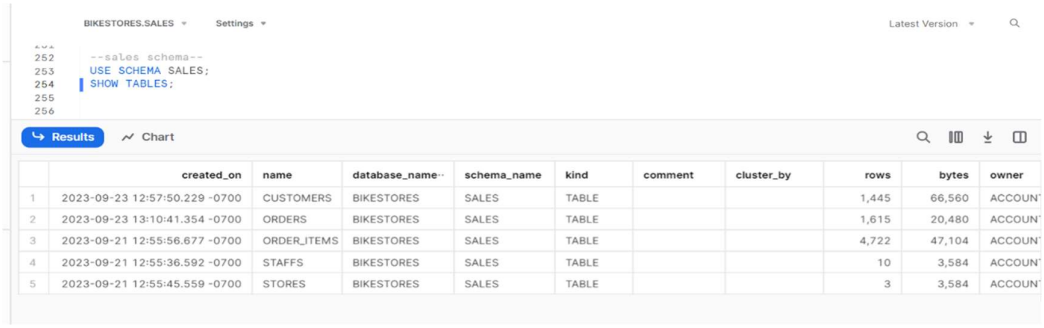


	created_on	name	database_name	schema_name	kind	comment	cluster_by	rows	bytes	owner
1	2023-09-21 12:56:31.526 -0700	BRANDS	BIKESTORES	PRODUCTION	TABLE			9	1,536	ACCOUN
2	2023-09-21 12:56:05.192 -0700	CATEGORIES	BIKESTORES	PRODUCTION	TABLE			7	1,536	ACCOUN
3	2023-09-21 12:56:13.079 -0700	PRODUCTS	BIKESTORES	PRODUCTION	TABLE			321	7,680	ACCOUN
4	2023-09-21 12:56:19.906 -0700	STOCKS	BIKESTORES	PRODUCTION	TABLE			939	4,096	ACCOUN

For Schema Sales:

USE SCHEMA SALES;

SHOW TABLES;

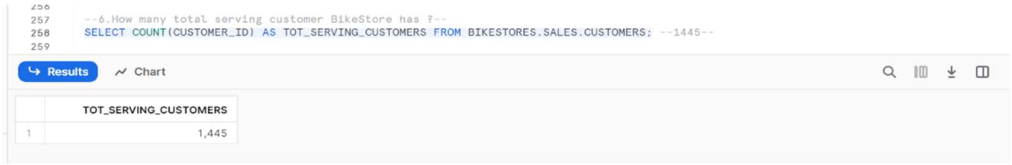


	created_on	name	database_name	schema_name	kind	comment	cluster_by	rows	bytes	owner
1	2023-09-23 12:57:50.229 -0700	CUSTOMERS	BIKESTORES	SALES	TABLE			1,445	66,560	ACCOUN
2	2023-09-23 13:10:41.354 -0700	ORDERS	BIKESTORES	SALES	TABLE			1,615	20,480	ACCOUN
3	2023-09-21 12:55:56.677 -0700	ORDER_ITEMS	BIKESTORES	SALES	TABLE			4,722	47,104	ACCOUN
4	2023-09-21 12:55:36.592 -0700	STAFFS	BIKESTORES	SALES	TABLE			10	3,584	ACCOUN
5	2023-09-21 12:55:45.559 -0700	STORES	BIKESTORES	SALES	TABLE			3	3,584	ACCOUN

6.How many total serving customer BikeStore has ?--

SELECT COUNT(CUSTOMER_ID) AS TOT_SERVING_CUSTOMERS
FROM BIKESTORES.SALES.CUSTOMERS;

total serving customers-1445



7.How many total orders are there ?--

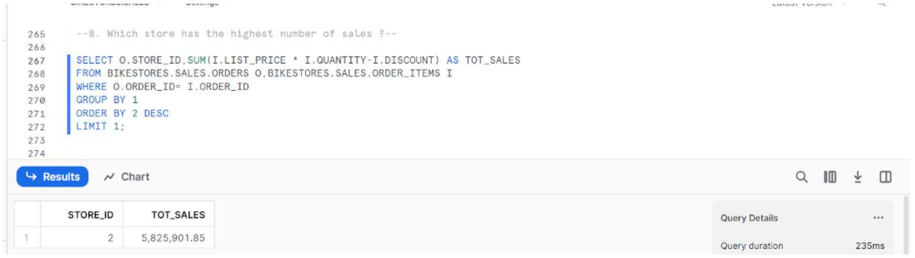
SELECT COUNT(ORDER_ID) FROM BIKESTORES.SALES.ORDERS;

Total orders -1615

8. Which store has the highest number of sales ?--

SELECT STORE_ID, COUNT(ORDER_ID) AS MAX_SALES
FROM BIKESTORES.SALES.ORDERS
GROUP BY STORE_ID
ORDER BY COUNT(ORDER_ID) DESC
LIMIT 1;

Store_ID 2 has highest number of sales



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

SELECT o.STORE_ID AS STORE_ID,
MONTHNAME(o.ORDER_DATE) AS `MONTH`,
SUM(i.LIST_PRICE * i.QUANTITY - i.DISCOUNT) AS TOT_SALES
FROM BIKESTORES.SALES.ORDERS o,BIKESTORES.SALES.ORDER_ITEMS i
WHERE o.ORDER_ID = i.ORDER_ID

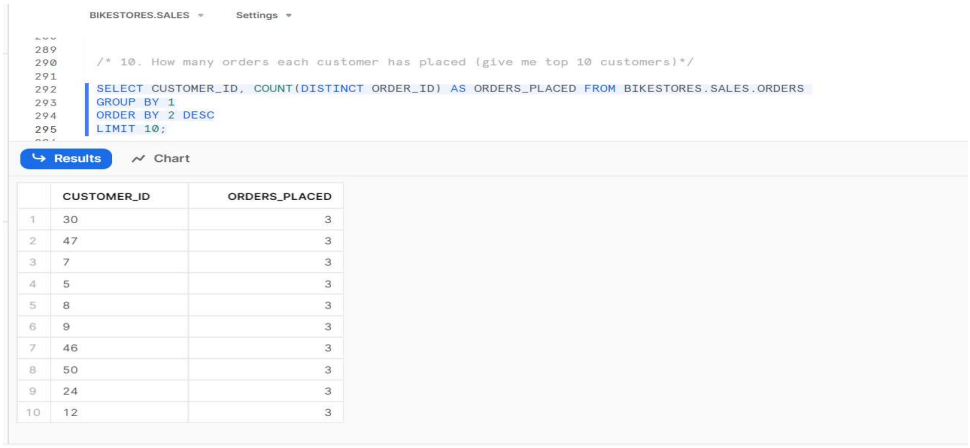
GROUP BY STORE_ID,`MONTH`
ORDER BY TOT_SALES DESC
LIMIT 1;

The Highest Sales is in April Which is 804633.1

---

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

SELECT CUSTOMER_ID, COUNT(DISTINCT ORDER_ID) AS ORDERS_PLACED
FROM BIKESTORES.SALES.ORDERS
GROUP BY 1
ORDER BY 2 DESC
LIMIT 10;



11.Which are the TOP 3 selling product ?

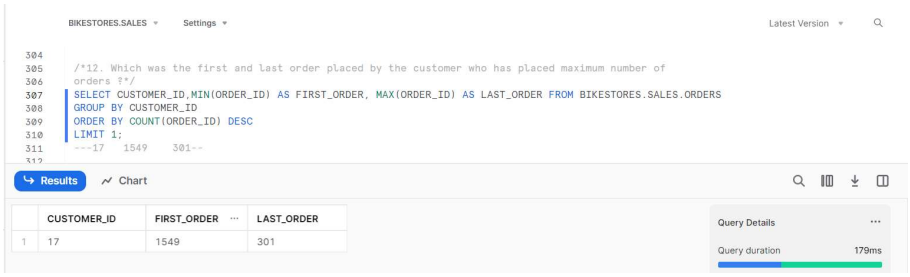
SELECT PRODUCT_ID,SUM(LIST_PRICE * QUANTITY - DISCOUNT) AS TOT_SALES
FROM BIKESTORES.SALES.ORDER_ITEMS
GROUP BY PRODUCT_ID

ORDER BY TOT_SALES DESC
LIMIT 3;

---

12. 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 CUSTOMER_ID
ORDER BY COUNT(ORDER_ID) DESC
LIMIT 1;



---

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

SELECT O.CUSTOMER_ID,
MIN(I.LIST_PRICE) AS CHEAPEST_PRODUCT_VALUE,
MAX(I.LIST_PRICE) AS COSTLIEST_PRODUCT_VALUE
FROM BIKESTORES.SALES.ORDERS O, BIKESTORES.SALES.ORDER_ITEMS I
WHERE O.ORDER_ID= I.ORDER_ID
GROUP BY 1
ORDER BY 2,3 DESC;

```
/*13. For every customer , which is the cheapest product and the costliest product which the
customer has bought.*/
SELECT  O.CUSTOMER_ID,
MIN(I.LIST_PRICE) AS CHEAPEST_PRODUCT_VALUE,
MAX(I.LIST_PRICE) AS COSTLIEST_PRODUCT_VALUE
FROM BIKESTORES.SALES.ORDERS O, BIKESTORES.SALES.ORDER_ITEMS I
WHERE O.ORDER_ID= I.ORDER_ID
GROUP BY 1
ORDER BY 2,3 DESC;
```

Results | Chart

	CUSTOMER_ID	CHEAPEST_PRODUCT_VALUE	COSTLIEST_PRODUCT_VALUE
1	1145	109.99	999.99
2	858	109.99	999.99
3	1229	109.99	999.99
4	939	109.99	999.99
5	192	109.99	899.99
6	730	109.99	832.99
7	114	109.99	799.99
8	110	109.99	749.99
9	120	109.99	749.99

CHEAPEST_PRODUCT_VALUE
1320.99

Here for first few records getting accurate output but then getting wrong output

---

14. 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 count(order_id)>=200
order by 2;

```
/*14. 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 count(order_id)>=200
order by 2;
```

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

ALTER TABLE BIKESTORES.SALES.ORDER_ITEMS ADD TOTAL_PRICE VARCHAR(50);

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=QUANTITY*LIST_PRICE;

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

ALTER TABLE BIKESTORES.SALES.ORDER_ITEMS ADD TOTAL_PRICE VARCHAR(50);

/*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=QUANTITY*LIST_PRICE;
```

Results | Chart

	number of rows updated	number of multi-joined rows updated
1	4,722	0

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

SELECT SUM(TOTAL_PRICE) FROM BIKESTORES.SALES.ORDER_ITEMS;

```
BIKESTORES.SALES    Settings

372   --17.What is the value of the TOTAL_PRICE paid for all the sales.order_items ? --
373   SELECT SUM(TOTAL_PRICE) FROM BIKESTORES.SALES.ORDER_ITEMS;
374
375
376
```

Results | Chart

	SUM(TOTAL_PRICE)
1	8,578,988.88

ER Diagram: