

## 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)  
);
```

DatabasesWorksheets

ACCOUNTADMINMY\_WAREHOUSEShare

Pinned (0)

No pinned objects

Q All Objects

Tables

BRANDS

CATEGORIES

PRODUCTS

STOCKS

PUBLIC

SALES

Tables

BOOKS1

BIKESTORES.SALES

Settings

Latest Version

Q

```

3  create database bikestores;
4  use bikestores;
5
6  CREATE OR REPLACE SCHEMA Sales;
7  CREATE OR REPLACE SCHEMA Production;
8  --table creation in sales--
9
10
11 CREATE OR REPLACE TABLE Sales.customers
12 (customer_id int AUTOINCREMENT,
13  first_name VARCHAR(50),
14  last_name VARCHAR(50),
15  phone VARCHAR(15),
16  email VARCHAR(60),
17  street VARCHAR(100),
18  city VARCHAR(50),
19  state VARCHAR(50),
20  zip_code VARCHAR(50),
21  PRIMARY KEY(customer_id)
22 );

```

DatabasesWorksheets

ACCOUNTADMINMY\_WAREHOUSEShare

Pinned (0)

No pinned objects

Q All Objects

Tables

BRANDS

CATEGORIES

PRODUCTS

STOCKS

PUBLIC

SALES

Tables

BOOKS1

BOOKS2

BOOK\_COMBINED

CUSTOMERS

ORDERS

BIKESTORES.SALES

Settings

Latest Version

Q

```

23
24 CREATE OR REPLACE TABLE Sales.orders
25 (
26  order_id VARCHAR(50) NOT NULL PRIMARY KEY,
27  customer_id VARCHAR(15),
28  order_status VARCHAR(50),
29  order_date DATE,
30  required_date DATE,
31  shipped_date DATE,
32  store_id INT,
33  staff_id INT
34 );
35
36 CREATE TABLE if not exists Sales.staffs
37 (
38  staff_id INT PRIMARY KEY,
39  first_name VARCHAR(50),
40  last_name VARCHAR(50),
41  phone VARCHAR(15),
42  email VARCHAR(60),
43  active VARCHAR(100),
44  store_id INT,
45  manager_id INT
46 );
47
48 CREATE TABLE if not exists Sales.stores
49 (
50  store_id int AUTOINCREMENT PRIMARY KEY,
51  store_name VARCHAR(50),
52  phone VARCHAR(15),
53  email VARCHAR(60),
54  street VARCHAR(100),
55  city VARCHAR(50),
56  state VARCHAR(50),
57  zip_code INT
58 );
59
60 CREATE TABLE if not exists Sales.order_items
61 (
62  order_id INT,
63  item_id INT,
64  product_id INT,
65  quantity VARCHAR(50),
66  list_price VARCHAR(50),
67  discount VARCHAR(50),
68  primary key(order_id, item_id)
69 );
70

```

DatabasesWorksheets

ACCOUNTADMINMY\_WAREHOUSEShare

Pinned (0)

No pinned objects

Q All Objects

Tables

BRANDS

CATEGORIES

PRODUCTS

STOCKS

PUBLIC

SALES

Tables

BOOKS1

BOOKS2

BOOK\_COMBINED

CUSTOMERS

ORDERS

ORDER\_ITEMS

STAFFS

BIKESTORES.SALES

Settings

Latest Version

Q

```

70
71 --CREATING PRODUCTION TABLE --
72
73 CREATE TABLE Production.categories
74 (
75  category_id INT PRIMARY KEY,
76  category_name VARCHAR(60)
77 );
78
79 CREATE TABLE Production.products
80 (
81  product_id INT PRIMARY KEY,
82  product_name VARCHAR(100),
83  brand_id INT,
84  category_id INT,
85  model_year INT,
86  list_price VARCHAR(50)
87 );
88
89 CREATE TABLE Production.stocks
90 (
91  store_id INT,
92  product_id INT,
93  quantity VARCHAR(50),
94  PRIMARY KEY(store_id, product_id)
95 );
96
97
98 CREATE TABLE if not exists Production.brands
99 (
100  brand_id INT PRIMARY KEY,
101  brand_name VARCHAR(100)
102 );
103

```

Results

Chart

Q

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 :

```

116  /*2. Once the table has got created , there is a requirement of FOREIGN KEY implementation coming
117  into picture where one needs to add(ALTER TABLE COMMAND) below foreign key on the table
118  mentioned pointing to another table (READ ABOUT FOREIGN KEY) as :*/
119
120  --sales.staffs (store_id) -> sales.stores(store_id)--
121  ALTER TABLE Sales.staffs ADD FOREIGN KEY(store_id) REFERENCES Sales.stores(store_id);
122
123  --sales.staffs (manager_id) -> sales.staffs (staff_id)--
124  ALTER TABLE Sales.staffs ADD FOREIGN KEY (manager_id) REFERENCES Sales.staffs(staff_id);
125
126  --production.products (category_id) -> production.categories (category_id)--
127  ALTER TABLE production.products ADD FOREIGN KEY (category_id) REFERENCES production.categories(category_id);
128
129  --production.products (brand_id) -> Production_brands (brand_id)
130  ALTER TABLE Production.products ADD FOREIGN KEY (brand_id) REFERENCES Production.brands(brand_id);
131
132  --sales.orders (customer_id) -> sales.customers (customer_id)
133  ALTER TABLE Sales.orders ADD FOREIGN KEY (customer_id) REFERENCES Sales.customers(customer_id);
134
135  --sales.orders (store_id) -> sales.stores (store_id)
136  ALTER TABLE Sales.orders ADD FOREIGN KEY (store_id) REFERENCES Sales.stores(store_id);
137
138  --sales.orders (staff_id) -> sales.staffs (staff_id)
139  ALTER TABLE Sales.orders ADD FOREIGN KEY (staff_id) REFERENCES Sales.staffs(staff_id);
140
141  --sales.order_items (order_id) -> sales.orders (order_id)
142  ALTER TABLE Sales.order_items ADD FOREIGN KEY (order_id) REFERENCES Sales.orders(order_id);
143
144  --sales.order_items (product_id) -> production.products (product_id)
145  ALTER TABLE Sales.order_items ADD FOREIGN KEY (product_id) REFERENCES Production.products(product_id);

```

Pinned (0)  
No pinned objects  
All Objects  
SALES  
Tables

BIKESTORES.SALES - Settings  

```

147  --production.stocks (store_id) -> sales.stores (store_id)
148  ALTER TABLE Production.stocks ADD FOREIGN KEY (store_id) REFERENCES Sales.stores(store_id);
149
150  --production.stocks (product_id) -> production.products (product_id)
151  ALTER TABLE Production.stocks ADD FOREIGN KEY (product_id) REFERENCES Production.products(product_id);
152

```

Latest Version

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)

Pinned (0)  
No pinned objects  
All Objects  
SALES  
Tables  
BOOKS1  
BOOKS2  
BOOK\_COMBINED  
CUSTOMERS

BIKESTORES.SALES - Settings  

```

160  /*3. Does any of the table has missing or NULL value ? If yes which are those and what are their
161  counts ?*/
162  --SALES SCHEMA--
163  SELECT COUNT(*) AS TOT_NULL_CUSTOMER FROM BIKESTORES.SALES.CUSTOMERS WHERE CUSTOMER_ID IS NULL OR FIRST_NAME IS NULL OR LAST_NAME IS
164  NULL OR PHONE IS NULL OR EMAIL IS NULL OR STREET IS NULL OR CITY IS NULL OR STATE IS NULL OR CITY IS NULL OR ZIP_CODE IS NULL;

```

Latest Version

Results Chart  

TOT_NULL_CUSTOMER
1 0

Query Details  
Query duration 99ms

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

BIKESTORES.SALES - Settings  

```

164  SELECT COUNT(*) AS TOT_NULL_ORDERS FROM BIKESTORES.SALES.ORDERS WHERE ORDER_ID IS NULL OR CUSTOMER_ID IS NULL OR ORDER_STATUS IS
165  NULL OR ORDER_DATE IS NULL OR REQUIRED_DATE IS NULL OR SHIPPED_DATE IS NULL OR STORE_ID IS NULL OR STAFF_ID IS NULL;

```

Latest Version

Results Chart  

TOT_NULL_ORDERS
1 170

Query Details  
Query duration 56ms

No null values present further..

```

171  SELECT COUNT(*) AS TOT_NULL_STAFFS FROM BIKESTORES.SALES.STAFFS WHERE FIRST_NAME IS NULL OR LAST_NAME IS NULL OR PHONE IS NULL OR
172  EMAIL IS NULL OR ACTIVE IS NULL;

```

Latest Version

Results Chart  

TOT_NULL_STAFFS
1 0

Query Details  
Query duration 58ms

```

167  SELECT COUNT(*) AS TOT_NULL_ORDER_ITEMS FROM BIKESTORES.SALES.ORDER_ITEMS WHERE ORDER_ID IS NULL OR ITEM_ID IS NULL OR PRODUCT_ID IS
168  NULL OR QUANTITY IS NULL OR LIST_PRICE IS NULL OR DISCOUNT IS NULL OR TOTAL_PRICE IS NULL;

```

Latest Version

Results Chart  

TOT_NULL_ORDER_ITEMS
1 0

Query Details  
Query duration 40ms

```
172 SELECT COUNT(*) AS TOT_NULL_STORES FROM BIKESTORES.SALES.STORES WHERE STORE_ID IS NULL OR STORE_NAME IS NULL OR PHONE IS NULL OR
173 EMAIL IS NULL OR STREET IS NULL OR CITY IS NULL OR STATE IS NULL OR ZIP_CODE IS NULL;
174
--PRODUCTION SCHEMA--
175
176 SELECT COUNT(*) AS TOT_NULL_BRANDS FROM BIKESTORES.PRODUCTION.BRANDS WHERE BRAND_ID IS NULL OR BRAND_NAME IS NULL;
177
178 SELECT COUNT(*) AS TOT_NULL_CATEGORIES FROM BIKESTORES.PRODUCTION.CATEGORIES WHERE CATEGORY_ID IS NULL OR CATEGORY_NAME IS NULL;
179
180 SELECT COUNT(*) AS TOT_NULL_PRODUCTS FROM BIKESTORES.PRODUCTION.PRODUCTS WHERE PRODUCT_ID IS NULL OR PRODUCT_NAME IS NULL OR
181 BRAND_ID IS NULL OR CATEGORY_ID IS NULL OR MODEL_YEAR IS NULL OR LIST_PRICE IS NULL;
182
183 SELECT COUNT(*) AS TOT_NULL_STOCKS FROM BIKESTORES.PRODUCTION.STOCKS WHERE STORE_ID IS NULL OR PRODUCT_ID IS NULL OR QUANTITY IS
184 NULL;
```

Results Chart

TOT_NULL_STORES
0

Query Details

Query duration: 36ms

Rows: 1

Results Chart

TOT_NULL_STOCKS
0

Query Details

Query duration: 59ms

Rows: 1

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 RECORDS.**

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--
202
203 SELECT COUNT(DISTINCT CUSTOMER_ID, FIRST_NAME, LAST_NAME, PHONE, EMAIL, STREET, CITY, STATE, ZIP_CODE) FROM
204 BIKESTORES.SALES.CUSTOMERS; --1445--
205
206
207
208 --ORDERS--
209 SELECT COUNT(DISTINCT ORDER_ID, CUSTOMER_ID, ORDER_STATUS, ORDER_DATE, REQUIRED_DATE, SHIPPED_DATE, STORE_ID, STAFF_ID) FROM
210 BIKESTORES.SALES.ORDERS; --1445--
211
212
213 --ORDERS_ITEMS--
214 SELECT COUNT(DISTINCT ORDER_ID, ITEM_ID, PRODUCT_ID, QUANTITY, LIST_PRICE, DISCOUNT) FROM BIKESTORES.SALES.ORDER_ITEMS; --4722--
215 SELECT COUNT(DISTINCT ORDER_ID, ITEM_ID, PRODUCT_ID, QUANTITY, LIST_PRICE, DISCOUNT) FROM BIKESTORES.SALES.ORDER_ITEMS; --4722--
216
217
218 --STAFF--
219 SELECT COUNT(DISTINCT STAFF_ID, FIRST_NAME, LAST_NAME, PHONE, EMAIL, ACTIVE, STORE_ID, MANAGER_ID) FROM BIKESTORES.SALES.STAFFS; --10--
220 SELECT COUNT(DISTINCT STAFF_ID, FIRST_NAME, LAST_NAME, PHONE, EMAIL, ACTIVE, STORE_ID, MANAGER_ID) FROM BIKESTORES.SALES.STAFFS; --10--
221
222
223 --STORES--
224 SELECT COUNT(DISTINCT STORE_ID, STORE_NAME, PHONE, EMAIL, STREET, CITY, STATE, ZIP_CODE) FROM BIKESTORES.SALES.STORES; --3--
225 SELECT COUNT(DISTINCT STORE_ID, STORE_NAME, PHONE, EMAIL, STREET, CITY, STATE, ZIP_CODE) FROM BIKESTORES.SALES.STORES; --3--
226
227
228 --BRANDS--
229 SELECT COUNT(DISTINCT BRAND_ID, BRAND_NAME) FROM BIKESTORES.PRODUCTION.BRANDS; --9--
230 SELECT COUNT(DISTINCT BRAND_ID, BRAND_NAME) FROM BIKESTORES.PRODUCTION.BRANDS; --9--
```

	BIKESTORES.SALES	Settings
229	--CATEGORIES--	
230	SELECT COUNT(CATEGORY_ID, CATEGORY_NAME) FROM BIKESTORES.PRODUCTION.CATEGORIES; --7--	
231	SELECT COUNT(DISTINCT CATEGORY_ID, CATEGORY_NAME) FROM BIKESTORES.PRODUCTION.CATEGORIES; --7--	
232		
233	--PRODUCTS--	
234	SELECT COUNT(PRODUCT_ID, PRODUCT_NAME, BRAND_ID, CATEGORY_ID, MODEL_YEAR, LIST_PRICE)	
235	FROM BIKESTORES.PRODUCTION.PRODUCTS; --321--	
236	SELECT COUNT(DISTINCT PRODUCT_ID, PRODUCT_NAME, BRAND_ID, CATEGORY_ID, MODEL_YEAR, LIST_PRICE)	
237	FROM BIKESTORES.PRODUCTION.PRODUCTS; --321--	
238		
239	--STOCKS--	
240	SELECT COUNT(STORE_ID, PRODUCT_ID, QUANTITY) FROM BIKESTORES.PRODUCTION.STOCKS; --939--	
241	SELECT COUNT(DISTINCT STORE_ID, PRODUCT_ID, QUANTITY) FROM BIKESTORES.PRODUCTION.STOCKS; --939--	
242		
243		

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;**

BIKESTORES.PRODUCTION

Settings

Latest Version

Q

245

/\*5.How many unique tables are present in each schema and under each table how many records are

246

we having ? (Write SQL Script for the same - I don't need answer like 3/5/4 etc)\*/

247

248

--production schema--

249

USE SCHEMA PRODUCTION;

250

SHOW TABLES;

251

Results

Chart

Q

📄

⬇

🗒

	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	ACCOUNT
2	2023-09-21 12:56:05.192 -0700	CATEGORIES	BIKESTORES	PRODUCTION	TABLE			7	1,536	ACCOUNT
3	2023-09-21 12:56:13.079 -0700	PRODUCTS	BIKESTORES	PRODUCTION	TABLE			321	7,680	ACCOUNT
4	2023-09-21 12:56:19.906 -0700	STOCKS	BIKESTORES	PRODUCTION	TABLE			939	4,096	ACCOUNT

For Schema Sales:

**USE SCHEMA SALES;**

**SHOW TABLES;**

BIKESTORES.SALES ▾

Settings ▾

252

--sales schema--

253

USE SCHEMA SALES;

254

SHOW TABLES;

255

256

Latest Version ▾

Q

Results

Chart

Q

🗑

⬇

📄

	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	ACCOUNT
2	2023-09-23 13:10:41.354 -0700	ORDERS	BIKESTORES	SALES	TABLE			1,615	20,480	ACCOUNT
3	2023-09-21 12:55:56.677 -0700	ORDER_ITEMS	BIKESTORES	SALES	TABLE			4,722	47,104	ACCOUNT
4	2023-09-21 12:55:36.592 -0700	STAFFS	BIKESTORES	SALES	TABLE			10	3,584	ACCOUNT
5	2023-09-21 12:55:45.559 -0700	STORES	BIKESTORES	SALES	TABLE			3	3,584	ACCOUNT

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

**SELECT COUNT(CUSTOMER\_ID) AS TOT\_SERVING\_CUSTOMERS**

**FROM BIKESTORES.SALES.CUSTOMERS;**

**total serving customers-1445**

```

256 --6.How many total serving customer BikeStore has ?--
257 SELECT COUNT(CUSTOMER_ID) AS TOT_SERVING_CUSTOMERS FROM BIKESTORES.SALES.CUSTOMERS; --1445--
258
259

```

TOT_SERVING_CUSTOMERS	
1	1,445

7.How many total orders are there ?--

**SELECT COUNT(ORDER\_ID) FROM BIKESTORES.SALES.ORDERS;**

**Total orders -1615**

```

259
260 --7.How many total orders are there ?--
261 SELECT COUNT(ORDER_ID) FROM BIKESTORES.SALES.ORDERS;--1615--
262
263
264

```

COUNT(ORDER_ID)	
1	1,615

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**

```

265 --8. Which store has the highest number of sales ?--
266
267 SELECT o.STORE_ID, SUM(i.LIST_PRICE * i.QUANTITY - i.DISCOUNT) AS TOT_SALES
268 FROM BIKESTORES.SALES.ORDERS o, BIKESTORES.SALES.ORDER_ITEMS i
269 WHERE o.ORDER_ID= i.ORDER_ID
270 GROUP BY 1
271 ORDER BY 2 DESC
272 LIMIT 1;
273
274

```

STORE_ID	TOT_SALES
2	5,825,901.85

Query duration 235ms

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**

```
--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;
--TOT SALES IN APRIL IS 804633.1 WHICH IS HIGHEST--
```

Results Chart

	...	STORE_ID	'MONTH'	TOT_SALES
1		2	Apr	804,633.1

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;**

BIKESTORES.SALES Settings

```
--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;
```

Results Chart

	CUSTOMER_ID	ORDERS_PLACED
1	30	3
2	47	3
3	7	3
4	5	3
5	8	3
6	9	3
7	46	3
8	50	3
9	24	3
10	12	3

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;**

```
296
297
298 /* 11. Which are the TOP 3 selling product ? */
299 SELECT PRODUCT_ID, SUM(LIST_PRICE * QUANTITY) AS TOT_SALES FROM BIKESTORES.SALES.ORDER_ITEMS
300 GROUP BY PRODUCT_ID
301 ORDER BY TOT_SALES DESC
302 LIMIT 3;
303
```

Results Chart

	PRODUCT_ID	TOT_SALES
1	7	615,998.46
2	9	434,998.55
3	4	414,698.57

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;**

```
304
305 /*12. Which was the first and last order placed by the customer who has placed maximum number of
306 orders ? */
307 SELECT CUSTOMER_ID, MIN(ORDER_ID) AS FIRST_ORDER, MAX(ORDER_ID) AS LAST_ORDER FROM BIKESTORES.SALES.ORDERS
308 GROUP BY CUSTOMER_ID
309 ORDER BY COUNT(ORDER_ID) DESC
310 LIMIT 1;
311 --17 1549 301--
312
```

Results Chart

	CUSTOMER_ID	FIRST_ORDER	LAST_ORDER
1	17	1549	301

Query Details  
Query duration 179ms

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;**

```

316
317
318 /*13. For every customer , which is the cheapest product and the costliest product which the
319 customer has bought */
320 SELECT O.CUSTOMER_ID,
321 MIN(I.LIST_PRICE) AS CHEAPEST_PRODUCT_VALUE,
322 MAX(I.LIST_PRICE) AS COSTLIEST_PRODUCT_VALUE
323 FROM BIKESTORES.SALES.ORDERS O, BIKESTORES.SALES.ORDER_ITEMS I
324 WHERE O.ORDER_ID= I.ORDER_ID
325 GROUP BY 1
326 ORDER BY 2,3 DESC;

```

	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

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;**

```

355
356 /*14. Which product has orders more than 200 ?*/
357
358 SELECT product_id,count(distinct order_id) as tot_orders FROM BIKESTORES.SALES.ORDER_ITEMS
359 GROUP BY 1
360 having count(order_id)>=200
361 order by 2;
362

```

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;**

```

363 /*15.Add a column TOTAL_PRICE with appropriate data type into the sales.order_items*/
364
365 ALTER TABLE BIKESTORES.SALES.ORDER_ITEMS ADD TOTAL_PRICE VARCHAR(50);
366
367 /*16.Calculate TOTAL_PRICE = quantity * list price and Update the value for all rows in the
368 sales.order_items table.*/
369
370 UPDATE BIKESTORES.SALES.ORDER_ITEMS SET TOTAL_PRICE=QUANTITY*LIST_PRICE;
371

```

	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:

