

```

1  CREATE DATABASE customer_shopping_data;
2  USE customer_shopping_data;
3
4  ----- import DATASET
5  SELECT * FROM customer_shopping_data;
6
7
8  -----
9  -- 1. Remove Duplicates
10 -- 2. Standarize the Data
11 -- 3. Null Values or blank values
12 -- 4. Remove Any Columns
13
14 -----
15 -- 1. Remove Duplicates
16
17 CREATE TABLE customer_shopping LIKE customer_shopping_data;
18
19 SELECT * FROM customer_shopping;
20
21 INSERT customer_shopping SELECT * FROM customer_shopping_data;
22
23 SELECT *,
24 ROW_NUMBER() OVER(
25 PARTITION BY invoice_no, customer_id,gender,age,category,quantity,price,payment_method,
invoice_date,shopping_mall) AS ROW_NUM
26 FROM customer_shopping;
27
28 SELECT * FROM customer_shopping;
29 SELECT * FROM customer_shopping_2;
30
31 CREATE TABLE `customer_shopping_2` (
32 `invoice_no` text,
33 `customer_id` text,
34 `gender` text,
35 `age` int DEFAULT NULL,
36 `category` text,
37 `quantity` int DEFAULT NULL,
38 `price` double DEFAULT NULL,
39 `payment_method` text,
40 `invoice_date` text,
41 `shopping_mall` text,
42 `row_num` INT
43 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
44
45
46 INSERT INTO customer_shopping_2
47 SELECT *,
48 ROW_NUMBER() OVER(
49 PARTITION BY invoice_no, customer_id,gender,age,category,quantity,price,payment_method,
invoice_date,shopping_mall) AS ROW_NUM
50 FROM customer_shopping;
51
52 -- Disable safe updates temporarily
53 SET SQL_SAFE_UPDATES = 0;
54
55 -- Delete duplicate rows based on row_num
56 DELETE FROM customer_shopping_2
57 WHERE row_num > 1;
58
59 -- Re-enable safe updates
60 SET SQL_SAFE_UPDATES = 1;
61
62 -- View final cleaned data
63 SELECT * FROM customer_shopping_2;
64
65
66 -----
67 -- 2. Standarizing Data

```

```

68 SELECT * FROM customer_shopping_2;
69
70 SELECT customer_id, TRIM(customer_id)
71 FROM customer_shopping_2;
72
73 UPDATE customer_shopping_2
74 SET customer_id = TRIM(customer_id);
75
76 SELECT payment_method, TRIM(payment_method) FROM customer_shopping_2;
77
78 SELECT DISTINCT invoice_no
79 FROM customer_shopping_2
80 ORDER BY 1;
81
82
83 SELECT `invoice_date`,
84 STR_TO_DATE(`invoice_date`, '%d/%m/%Y')
85 FROM customer_shopping_2;
86
87 UPDATE customer_shopping_2
88 SET invoice_date = STR_TO_DATE(invoice_date, '%d/%m/%Y');
89
90
91 -- UPDATE price decimal TO ROUND NUMBER.
92
93 UPDATE customer_shopping_2
94 SET price = ROUND(price, 0);
95
96
97 ALTER TABLE customer_shopping_2
98 MODIFY COLUMN `invoice_date` DATE;
99
100 -----
101 -- 3. Null Values or blank values of Eevery Columns
102
103 SELECT * FROM customer_shopping_2;
104
105 SELECT DISTINCT category
106 FROM customer_shopping_2
107 WHERE category IS NULL
108 OR category = '';
109
110 SELECT DISTINCT customer_id
111 FROM customer_shopping_2
112 WHERE customer_id IS NULL
113 OR customer_id = '';
114
115 SELECT DISTINCT invoice_no
116 FROM customer_shopping_2
117 WHERE invoice_no IS NULL
118 OR invoice_no = '';
119
120 SELECT DISTINCT age
121 FROM customer_shopping_2
122 WHERE age IS NULL
123 OR age = '';
124
125 SELECT DISTINCT payment_method
126 FROM customer_shopping_2
127 WHERE payment_method IS NULL
128 OR payment_method = '';
129
130 SELECT DISTINCT price
131 FROM customer_shopping_2
132 WHERE price IS NULL
133 OR price = '';
134
135 SELECT DISTINCT invoice_date
136 FROM customer_shopping_2

```

```

137 WHERE invoice_date IS NULL
138 OR invoice_date = '';
139
140 SELECT DISTINCT shopping_mall
141 FROM customer_shopping_2
142 WHERE shopping_mall IS NULL
143 OR shopping_mall = '';
144
145 SELECT DISTINCT shopping_mall
146 FROM customer_shopping_2
147 WHERE shopping_mall IS NULL
148 OR shopping_mall = '';
149
150
151 -- CHecking the Datatypes
152 DESC customer_shopping_2;
153 ---
154
155 ALTER TABLE customer_shopping_2
156 DROP COLUMN row_num;
157
158 SELECT * FROM customer_shopping_2;
159
160 -- EDA DATA ANALYSIS AND DATA PRE-PROCESSING
161
162 SELECT MAX(price)
163 FROM customer_shopping_2;
164
165 SELECT MAX(price)
166 FROM customer_shopping_2
167 WHERE price = 5250
168 ORDER BY customer_id DESC ;
169
170 -----
171 -- Total Price According customer_id
172
173 SELECT customer_id, ROUND(SUM(price),0) AS SUM_OF_PRICE
174 FROM customer_shopping_2
175 GROUP BY customer_id
176 ORDER BY 2 DESC;
177
178 -----
179 -- Total Price According MIN & MAX DATE
180
181 SELECT MIN(`invoice_date`), MAX(`invoice_date`), SUM(price)
182 FROM customer_shopping_2;
183
184 -----
185 -- Total Price According category
186
187 SELECT category, ROUND(SUM(price),0)
188 FROM customer_shopping_2
189 GROUP BY category
190 ORDER BY 2 DESC;
191
192 -----
193 -- Total Price According shopping_mall
194
195 SELECT shopping_mall, ROUND(SUM(price),0)
196 FROM customer_shopping_2
197 GROUP BY shopping_mall
198 ORDER BY 2 DESC;
199
200 -----
201 -- Total Price According DISTINCT YEAR
202
203 SELECT YEAR(`invoice_date`), ROUND(SUM(price),0)
204 FROM customer_shopping_2
205 GROUP BY YEAR(`invoice_date`)

```

```

206 ORDER BY 1 DESC;
207 -----
208 -- Total Price According the MONTH with Date
209
210 SELECT SUBSTRING(`invoice_date`, 6,2) AS 'MONTH',SUBSTRING(`invoice_date`, 9, 2) as dat,
ROUND(SUM(price),0)
211 FROM customer_shopping_2
212 WHERE SUBSTRING(`invoice_date`, 6,2) IS NOT NULL
213 GROUP BY SUBSTRING(`invoice_date`, 6,2), SUBSTRING(`invoice_date`, 9, 2)
214 ORDER BY 1 ASC;
215
216 -----
217 -- Total Price According the MONTH WITH YEAR
218
219 SELECT SUBSTRING(`invoice_date`,1,7) AS 'MONTH', ROUND(SUM(price),0)
220 FROM customer_shopping_2
221 WHERE SUBSTRING(`invoice_date`, 1,7) IS NOT NULL
222 GROUP BY SUBSTRING(`invoice_date`, 1,7)
223 ORDER BY 1 ASC;
224
225 -- Total Price According the shopping_mall WITH YEAR
226
227 SELECT DISTINCT shopping_mall, YEAR(`invoice_date`), ROUND(SUM(price),0) AS SUM_OF_PRICE
228 FROM customer_shopping_2
229 GROUP BY shopping_mall, YEAR(`invoice_date`)
230 ORDER BY 3 DESC;
231
232
233 SELECT * FROM customer_shopping_2;
234
235
236
237

```