

SQL Code

```
1  use my_db;
2
3  -- Verify data insertion
4  SELECT COUNT(*) AS total_records FROM retail_sales;
5  SELECT * FROM retail_sales LIMIT 10;
6
7
8  -- Checked for NULL values across all columns.
9  SELECT * FROM retail_sales
10 WHERE transactions_id IS NULL;
11
12 SELECT * FROM retail_sales
13 WHERE sale_time IS NULL;
14
15 SELECT * FROM retail_sales
16 WHERE sale_date IS NULL;
17
18 SELECT * FROM retail_sales
19 WHERE customer_id IS NULL;
20
21 SELECT * FROM retail_sales
22 WHERE gender IS NULL;
23
24 SELECT * FROM retail_sales
25 WHERE age IS NULL;
26
27 SELECT * FROM retail_sales
28 WHERE category IS NULL;
29
30 SELECT * FROM retail_sales
31 WHERE quantity IS NULL;
32
33 SELECT * FROM retail_sales
34 WHERE cogs IS NULL;
35
36 SELECT * FROM retail_sales
37 WHERE total_sale IS NULL;
38
39 -- Removed incomplete records to ensure data integrity.
40 SELECT * FROM retail_sales
41 WHERE transactions_id IS NULL
42     OR sale_date IS NULL
43     OR sale_time IS NULL
44     OR customer_id IS NULL
45     OR gender IS NULL
46     OR age IS NULL
47     OR category IS NULL
48     OR quantity IS NULL
49     OR price_per_unit IS NULL
50     OR cogs IS NULL
51     OR total_sale IS NULL;
52
```

```

53  set sql_safe_updates = 0;
54
55  DELETE FROM retail_sales
56  WHERE transactions_id IS NULL
57         OR sale_date IS NULL
58         OR sale_time IS NULL
59         OR customer_id IS NULL
60         OR gender IS NULL
61         OR age IS NULL
62         OR category IS NULL
63         OR quantity IS NULL
64         OR price_per_unit IS NULL
65         OR cogs IS NULL
66         OR total_sale IS NULL;
67
68
69  -- Exploratory Data Analysis (EDA)
70  /*
71  Q1. Total Records Count
72  */
73  SELECT COUNT(*) AS TotalRecords
74  FROM retail_sales;
75
76  /*
77  Q2. Count of unique customers
78  */
79  SELECT COUNT(DISTINCT customer_id) AS TotalCustomers
80  FROM retail_sales;
81
82
83  /*
84  Q3. HOW MANY CATEGORIES DO WE HAVE ?
85  */
86  SELECT COUNT(DISTINCT Category) AS TotalCategories
87  FROM retail_sales;
88
89
90  -- DATA ANALYSIS OVER BUSINESS KEY PROBLEM
91  /*
92  Q1. Write a SQL query to retrieve all columns for sales made on '2022-11-05:
93  */
94  SELECT
95      *
96  FROM retail_sales
97  WHERE sale_date = '2022-11-05';
98
99  /*
100  Q2. Write a SQL query to retrieve all transactions where the category is 'Clothing' a
nd the quantity sold is more than 3 in the month of Nov-2022
101  */
102
103  SELECT
104      *
105  FROM retail_sales
106  WHERE category = 'Clothing'

```

```

107         AND
108         quantity > 3
109     AND
110     sale_date BETWEEN '2022-11-01' AND '2022-11-30';
111
112  /*
113  Q3. Write a SQL query to calculate the total sales (total_sale) for each category.
114  */
115  SELECT
116      category,
117      SUM(total_sale) AS TotalSales
118  FROM retail_sales
119  GROUP BY
120      category
121  ORDER BY
122      TotalSales DESC;
123
124  SELECT
125      category,
126      count(category) AS totalquantity
127  FROM retail_sales
128  GROUP BY category;
129
130  /*
131  Q4. Write a SQL query to find the average age of customers who purchased items from
the 'Beauty' category.
132  */
133
134  SELECT
135      category,
136      ROUND(AVG(age)) AS AverageAge
137  FROM retail_sales
138  WHERE category = 'Beauty';
139
140  /*
141  Q5. Write a SQL query to find all transactions where the total_sale is greater than
1000.
142  */
143  SELECT
144      transactions_id,
145      total_sale
146  FROM retail_sales
147  WHERE total_sale > 1000;
148
149
150  /*
151  Q6. Write a SQL query to find the total number of transactions (transaction_id) made
by each gender in each category.
152  */
153  SELECT
154      gender,
155      category,
156      COUNT(*) AS TotalTransactions
157  FROM retail_sales
158  GROUP BY 1,2

```

```

159 ORDER BY 2;
160
161
162 /*
163 Q7. Write a SQL query to calculate the average sale for each month. Find out best sel
ling month in each year:
164 */
165 SELECT
166     Year,
167     Month,
168     AverageSale
169 FROM (
170     SELECT
171         YEAR(sale_date) AS Year,
172         MONTH(sale_date) AS Month,
173         ROUND(AVG(total_sale),2) AS AverageSale,
174         DENSE_RANK() OVER(PARTITION BY YEAR(sale_date) ORDER BY AVG(total_sale)
DESC) AS AverageRank
175     FROM retail_sales
176     GROUP BY 1,2
177 )t
178 WHERE AverageRank = 1;
179
180
181 /*
182 Q8. Write a SQL query to find the top 5 customers based on the highest total sales:
183 */
184 SELECT
185     customer_id,
186     COUNT(*) AS Purchase,
187     SUM(total_sale) AS TotalSales
188 FROM retail_sales
189 GROUP BY
190     customer_id
191 ORDER BY
192     TotalSales DESC
193 LIMIT 5;
194
195 SELECT
196     *
197 FROM retail_sales
198 WHERE customer_id = 1003;
199
200
201 -- Beauty category attracts younger customers
202 SELECT category,
203     ROUND(AVG(age), 1) as avg_age,
204     MIN(age) as min_age,
205     MAX(age) as max_age
206 FROM retail_sales
207 GROUP BY category
208 ORDER BY avg_age;
209
210
211 /*

```

212 Q9. Write a SQL query to find the number of unique customers who purchased items from each category:

```
213 */
214 SELECT
215     category,
216     COUNT(DISTINCT(customer_id)) AS unique_customers
217 FROM retail_sales
218 GROUP BY category;
```

219

220

221 /*

222 Q10. Write a SQL query to create each shift and number of orders

223 (Example Morning <12, Afternoon Between 12 & 17, Evening >17):

224 */

225

226 WITH hourly_sale

227 AS

228 (

229 SELECT

230 *,

231 CASE

232 WHEN HOUR(sale_time) < 12 THEN "Morning"

233 WHEN HOUR(sale_time) BETWEEN 12 AND 17 THEN "Afternoon"

234 ELSE "Evening"

235 END AS Shift

236 FROM retail_sales

237)

238 SELECT

239 Shift,

240 COUNT(*) AS TotalOrders

241 FROM hourly_sale

242 GROUP BY Shift;