

SCHEMAS

Filter objects

- smithazon
- sys
- terpbuy
 - Tables
 - Views
 - Stored Proc
 - Functions

Administrati

Information

No object selected

Limit to 1000 rows

```

1  -- Query: Create the Database
2  -- Purpose:
3  -- Creates the `terpbuy` database and sets it as active.
4  • CREATE DATABASE IF NOT EXISTS terpbuy;
5  • USE terpbuy;

```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
2	16:36:02	USE terpbuy	0 row(s) affected	0.000 sec
3	16:36:52	CREATE TABLE department (department_id INT PRIM...	Error Code: 1050. Table 'department' already exists	0.031 sec
4	16:37:39	CREATE DATABASE IF NOT EXISTS terpbuy	1 row(s) affected, 1 warning(s): 1007 Can't create database ...	0.000 sec
5	16:37:40	USE terpbuy	0 row(s) affected	0.000 sec

Object Info

SCHEMAS

Filter objects

- smithazon
- sys
- terpbuy
 - Tables
 - Views
 - Stored Proc
 - Functions

Administrati

Information

No object selected

Limit to 1000 rows

```

8  -- Purpose:
9  -- Creates the `department` table to store department information.
10
11 • CREATE TABLE department (
12     department_id INT PRIMARY KEY,
13     department_name VARCHAR(20)
14 );
15
16 • SHOW TABLES;
17

```

Result Grid

Filter Rows:

Exports

Wrap Cell Contents

Tables_in_terpbuy
category
customer
department
order_line
orders
product

Result 1 x

Read Only

```

18  -- Query: Create the `category` Table
19  -- Purpose:
20  -- Creates the `category` table to store product category information.
21
22  • CREATE TABLE category (
23      category_id INT PRIMARY KEY,
24      category_name VARCHAR(50)
25  );

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Tables_in_terpbuy
▶ category
customer
department
order_line
orders
product

```

--
29  -- Query: Create the `customer` Table
30  -- Purpose:
31  -- Creates the `customer` table to store customer details.
--

```

```

33  • CREATE TABLE customer (
34      customer_id INT PRIMARY KEY,
35      first_name VARCHAR(50),
36      last_name VARCHAR(50),
37      street VARCHAR(100),
38      city VARCHAR(50),
39      state VARCHAR(50),
40      zip_code VARCHAR(10),
41      segment VARCHAR(50)
42  );

```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

Tables_in_terpbuy
category
customer
department
order_line
orders
product

```

84  -- Query: Create the `order_line` Table
85  -- Purpose:
86  -- Creates the `order_line` table to store individual order item details,
87  -- with foreign keys linking it to the `orders` and `product` tables.
--
89  • CREATE TABLE order_line (
90      order_line_id INT PRIMARY KEY,
91      order_id INT,
92      product_id INT,
93      quantity_sold INT,
94      total_price DECIMAL(10, 2),
95      FOREIGN KEY (order_id) REFERENCES orders(order_id),
96      FOREIGN KEY (product_id) REFERENCES product(product_id)
97  );
98
99  • SHOW TABLES;

```

100

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

Tables_in_terpbuy
category
customer
department
order_line
orders
product

Result 4 ×

```
101 -- Query: Describe the `orders` Table
102 -- Purpose:
103 -- Displays the structure of the `orders` table.
104
105 • DESCRIBE orders;
106
107 -- Query: Select Limited Rows from `orders`
```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	Field	Type	Null	Key	Default	Extra
▶	order_id	int	NO	PRI	NULL	
	customer_id	int	YES	MUL	NULL	
	order_date	date	YES		NULL	
	order_status	varchar(50)	YES		NULL	
	shipping_mode	varchar(50)	YES		NULL	
	order_city	varchar(50)	YES		NULL	
	order_region	varchar(50)	YES		NULL	
	scheduled_shipping_days	int	YES		NULL	
	actual_shipping_days	int	YES		NULL	
	payment_type	varchar(50)	YES		NULL	

```
107 -- Query: Select Limited Rows from `orders`
108 -- Purpose:
109 -- Retrieves up to 5 rows from the `orders` table to confirm data.
110
111 • SELECT * FROM orders LIMIT 5;
112
113 -- Query: Retrieve a Customer by ID
114 -- Purpose:
115 -- Retrieves details of the customer with `customer_id = 1`.
116
```

[illegible]

```

113 -- Query: Retrieve a Customer by ID
114 -- Purpose:
115 -- Retrieves details of the customer with `customer_id = 1`.
116
117 • SELECT * FROM customer WHERE customer_id = 1;
118
119 • DESCRIBE orders;
120
121 • DELETE FROM orders WHERE order_id = 1;
122

```

<div> <div>Result Grid</div> <div>Filter Rows:</div> <div>Edit:</div> <div>Export/Import:</div> <div>Wrap Cell Content:</div> </div>								
	customer_id	first_name	last_name	street	city	state	zip_code	segment
▶	1	John	Doe	123 Main St	Los Angeles	California	90001	Consumer
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```

126 -- Query: Check Foreign Key Constraints in `orders`
127 -- Purpose:
128 -- Lists foreign key constraints for the `orders` table,
129 -- showing which columns reference other tables and columns.
130

```

```
130
131 • SELECT
132     TABLE_NAME, COLUMN_NAME, CONSTRAINT_NAME, REFERENCED_TABLE_NAME, REFERENCED_COLUMN_NAME
133 FROM
134     INFORMATION_SCHEMA.KEY_COLUMN_USAGE
135 WHERE
136     TABLE_NAME = 'orders' AND TABLE_SCHEMA = 'terpbuy';
137
138 • SELECT * FROM customer WHERE customer_id = 1;
139
140 -- Query: Insert Data into the `customer` Table
```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	TABLE_NAME	COLUMN_NAME	CONSTRAINT_NAME	REFERENCED_TABLE_NAME	REFERENCED_COLUMN_NAME
▶	orders	order_id	PRIMARY	<small>HULL</small>	<small>HULL</small>
	orders	customer_id	orders_ibfk_1	customer	customer_id

```
140 -- Query: Insert Data into the `customer` Table
141 -- Purpose:
142 -- Adds a sample customer record to the `customer` table.
143
144 • INSERT INTO customer (customer_id, first_name, last_name, street, city, state, zip_code, segment)
145 VALUES (1, 'John', 'Doe', '123 Main St', 'Los Angeles', 'California', '90001', 'Consumer');
146
```

```
147 -- Query: Check Foreign Key Constraints in the `orders` Table
148 -- Purpose:
149 -- Displays the foreign key relationships of the `orders` table.
150
151 • SELECT
152     TABLE_NAME, COLUMN_NAME, CONSTRAINT_NAME, REFERENCED_TABLE_NAME, REFERENCED_COLUMN_NAME
153 FROM
154     INFORMATION_SCHEMA.KEY_COLUMN_USAGE
155 WHERE
156     TABLE_NAME = 'orders' AND TABLE_SCHEMA = 'terpbuy';
157
```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	TABLE_NAME	COLUMN_NAME	CONSTRAINT_NAME	REFERENCED_TABLE_NAME	REFERENCED_COLUMN_NAME
▶	orders	order_id	PRIMARY	<small>HULL</small>	<small>HULL</small>
	orders	customer_id	orders_ibfk_1	customer	customer_id

KEY COLUMN USAGE 10

```

162 -- Query: Insert Data into the `orders` Table
163 -- Purpose:
164 -- Adds a sample order record to the `orders` table.
165
166 • INSERT INTO orders (order_id, customer_id, order_date, order_status, shipping_mode, order_city, order_reg
167 VALUES (1, 1, '2024-11-29', 'Completed', 'First Class', 'Los Angeles', 'West', 3, 2);
168
169 -- Query: Insert Data into the `department` Table
170 -- Purpose:
171 -- Adds sample records to the `department` table for different business departments.
172
173 • INSERT INTO department (department_id, department_name)
174 VALUES
175 (1, 'Electronics'),
176 (2, 'Clothing'),
177 (3, 'Home & Kitchen');
178
179 • SELECT * FROM department;
180
181 -- Query: Insert Data into the `category` Table

```

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
department_id	department_name			
1	Electronics			
2	Clothing			
3	Home & Kitchen			
NULL	NULL			

```

181 -- Query: Insert Data into the `category` Table
182 -- Purpose:
183 -- Adds sample records to the `category` table for different product categories.
184
185 • INSERT INTO category (category_id, category_name)
186 VALUES
187 (1, 'Mobile Phones'),
188 (2, 'Laptops'),
189 (3, 'T-Shirts'),
190 (4, 'Cookware'),
191 (5, 'Furniture');

```

```

193 • SELECT * FROM category;
194

```

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
category_id	category_name			
1	Mobile Phones			
2	Laptops			
3	T-Shirts			
4	Cookware			
5	Furniture			
NULL	NULL			

```

195 -- Query: Insert Data into the `product` Table
196 -- Purpose:
197 -- Adds sample records to the `product` table for different products with categories and prices.
198
199 • INSERT INTO product (product_id, product_name, category_id, dept_id, product_price)
200 VALUES
201 (1, 'iPhone 15', 1, 1, 999.99),
202 (2, 'MacBook Air', 2, 1, 1249.99),
203 (3, 'Graphic T-Shirt', 3, 2, 19.99),
204 (4, 'Non-Stick Pan', 4, 3, 49.99),
205 (5, 'Sofa Set', 5, 3, 899.99);

```

```

207 • SELECT * FROM product;
208

```

Result Grid					
Filter Rows:		Edit:		Export/Import:	
product_id	product_name	category_id	dept_id	product_price	
1	iPhone 15	1	1	999.99	
2	MacBook Air	2	1	1249.99	
3	Graphic T-Shirt	3	2	19.99	
4	Non-Stick Pan	4	3	49.99	
5	Sofa Set	5	3	899.99	
•	NULL	NULL	NULL	NULL	

```

210 -- Query: Insert Data into the `order_line` Table
211 -- Purpose:
212 -- Adds sample records to the `order_line` table for products included in specific orders.
213
214 • INSERT INTO order_line (order_line_id, order_id, product_id, quantity_sold, total_price)
215 VALUES
216 (1, 1, 1, 2, 1999.98), -- 2 iPhones for order 1
217 (2, 1, 3, 3, 59.97), -- 3 T-Shirts for order 1
218 (3, 1, 5, 1, 899.99); -- 1 Sofa Set for order 1

```

```

220 • SELECT * FROM order_line;
221

```

Result Grid					
Filter Rows:		Edit:		Export/Import:	
order_line_id	order_id	product_id	quantity_sold	total_price	
1	1	1	2	1999.98	
2	1	3	3	59.97	
3	1	5	1	899.99	
•	NULL	NULL	NULL	NULL	

```

223 -- Query: View All Orders with Customer Details
224 -- Purpose:
225 -- Retrieves details about orders and their associated customers
226 -- by joining the `orders` and `customer` tables.
227

```






```

228 • SELECT
229     orders.order_id,
230     orders.order_date,
231     orders.order_status,
232     customer.first_name,
233     customer.last_name,
234     customer.city,
235     customer.state

236 FROM
237     orders
238 JOIN
239     customer
240 ON
241     orders.customer_id = customer.customer_id;
242

```

Result Grid							
Filter Rows: <input type="text"/>							
Exports:  Wrap Cell Content: 							
	order_id	order_date	order_status	first_name	last_name	city	state
▶	1	2024-11-29	Completed	John	Doe	Los Angeles	California

Result 16 x 

```

243 -- Query: Calculate Total Quantity Sold and Revenue per Product
244 -- Purpose:
245 -- Aggregates data to calculate the total quantity sold and total revenue for each product.
246
247 • SELECT
248     product.product_name,
249     SUM(order_line.quantity_sold) AS total_quantity_sold,
250     SUM(order_line.total_price) AS total_revenue

```

```

251 FROM
252     order_line
253 JOIN
254     product
255 ON
256     order_line.product_id = product.product_id
257 GROUP BY
258     product.product_name;
259
260 -- Query: Calculate Total Quantity Sold and Revenue per Product
261 -- Purpose:

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
product_name	total_quantity_sold	total_revenue	
▶ iPhone 15	2	1999.98	
Graphic T-Shirt	3	59.97	
Sofa Set	1	899.99	

```

260 -- Query: Calculate Total Quantity Sold and Revenue per Product
261 -- Purpose:
262 -- Aggregates data to calculate the total quantity sold and total revenue for each product.
263

```

```

264 • SELECT
265     product.product_name,
266     SUM(order_line.quantity_sold) AS total_quantity_sold,
267     SUM(order_line.total_price) AS total_revenue
268 FROM
269     order_line
270 JOIN
271     product
272 ON
273     order_line.product_id = product.product_id
274
275 ON
276     order_line.product_id = product.product_id
277 GROUP BY
278     product.product_name;
279

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
product_name	total_quantity_sold	total_revenue	
▶ iPhone 15	2	1999.98	
Graphic T-Shirt	3	59.97	
Sofa Set	1	899.99	

```

276
277 -- Query: Calculate Total Revenue per Product Category
278 -- Purpose:
279 -- Aggregates data to calculate the total revenue generated for each product category.
280
281 • SELECT
282     category.category_name,
283     SUM(order_line.total_price) AS total_revenue
284 FROM
285     order_line

```

```

JOIN
    product
ON
    order_line.product_id = product.product_id
JOIN
    category
ON
    product.category_id = category.category_id

```

```

294 GROUP BY
295     category.category_name;
296
297 -- Query: Calculate Average Shipping Delay by Region
298 -- Purpose:

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
category_name	total_revenue		
Mobile Phones	1999.98		
T-Shirts	59.97		
Furniture	899.99		

```

297 -- Query: Calculate Average Shipping Delay by Region
298 -- Purpose:
299 -- Calculates the average difference between actual and scheduled shipping days for each region.
300

```

```

301 • SELECT
302     order_region,
303     AVG(actual_shipping_days - scheduled_shipping_days)
304 FROM
305     orders
306 GROUP BY
307     order_region;

```

```

309 -- Query: Calculate Total Orders and Revenue by Customer Segment
310 -- Purpose:
311 -- Aggregates data to calculate the total number of distinct orders and total revenue
312 -- for each customer segment.

```

```

313
314 • SELECT
315     customer.segment,
316     COUNT(DISTINCT orders.order_id) AS total_orders,
317     SUM(order_line.total_price) AS total_revenue
318 FROM
319     orders
320 JOIN
321     customer

```

```

320 JOIN
321     customer
322 ON
323     orders.customer_id = customer.customer_id
324 JOIN
325     order_line
326 ON
327     orders.order_id = order_line.order_id
328 GROUP BY
329     customer.segment;
330

```

Result Grid			
	segment	total_orders	total_revenue
▶	Consumer	1	2959.94