

TerpBuy: Project Report

MySQL analysis

By: Vishesh Dabas

Problem Statement #1

How many rows of data are stored for each table in the database? List the name of each table followed by the number of rows it has.

Query #1

```
select 'Category' as 'Tablename', count(*) as 'Row Count' from category union all
select 'Customer' as 'Tablename', count(*) as 'Row Count' from customer union all
select 'Department' as 'Tablename', count(*) as 'Row Count' from department union all
select 'Order Line' as 'Tablename', count(*) as 'Row Count' from order_line union all
select 'Orders' as 'Tablename', count(*) as 'Row Count' from orders union all
select 'Product' as 'Tablename', count(*) as 'Row Count' from product;
```



The screenshot shows a MySQL query editor with the following SQL query:

```
-- ans1 Vishesh dabas - 18/09/2023
select 'Category' as 'Tablename', count(*) as 'Row Count' from category union all
select 'Customer' as 'Tablename', count(*) as 'Row Count' from customer union all
select 'Department' as 'Tablename', count(*) as 'Row Count' from department union all
select 'Order Line' as 'Tablename', count(*) as 'Row Count' from order_line union all
select 'Orders' as 'Tablename', count(*) as 'Row Count' from orders union all
select 'Product' as 'Tablename', count(*) as 'Row Count' from product;
```

Below the query editor, the results are displayed in a table grid. The table has two columns: 'Tablename' and 'Row Count'.

Tablename	Row Count
Category	51
Customer	4461
Department	12
Order Line	4783
Orders	2152
Product	72

Problem Statement #2

Which products are considered high-priced products? A high-priced product has a price exceeding \$100.00. List the names and prices of the high-priced products.

Query #2

```
select product_name, product_price from product where product_price > 100 order by  
product_price desc ;
```

```
10  -- ans 2 Vishesh dabas - 18/09/2023
11  • select product_name, product_price from product where product_price > 100 order by product_price desc ;
12  |
```

product_name	product_price
Dell Laptop	1500.00
Lawn mower	532.58
Porcelain crafts	461.48
Web Camera	452.04
Field & Stream Sportsman 16 Gun Fire Safe	399.98
Childrens heaters	357.10
Smart watch	327.75
Diamondback Womens Serene Classic Comfort Bi	299.98
First aid kit	293.04
Rock music	260.65
Industrial CONSUMER electronics	252.88

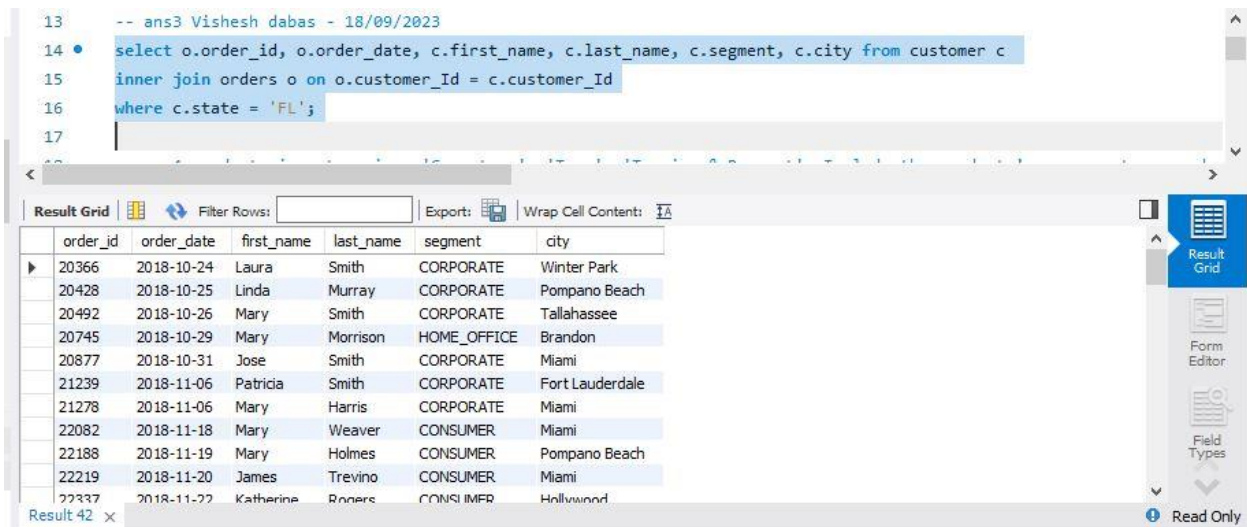
product 41 x

Problem Statement #3

List all orders placed by customers in the state of Florida. Note: The state abbreviation for Florida is 'FL'. Include the customers' first names, last names, city, and segment, along with the order ID and order date.

Query #3

```
select o.order_id, o.order_date, c.first_name, c.last_name, c.segment, c.city from customer c
inner join orders o on o.customer_Id = c.customer_Id
where c.state = 'FL';
```



The screenshot shows a database query editor with the following SQL query:

```
-- ans3 Vishesh dabas - 18/09/2023
select o.order_id, o.order_date, c.first_name, c.last_name, c.segment, c.city from customer c
inner join orders o on o.customer_Id = c.customer_Id
where c.state = 'FL';
```

Below the query, the results are displayed in a table grid. The table has 7 columns: order_id, order_date, first_name, last_name, segment, and city. The results show 12 rows of data.

order_id	order_date	first_name	last_name	segment	city
20366	2018-10-24	Laura	Smith	CORPORATE	Winter Park
20428	2018-10-25	Linda	Murray	CORPORATE	Pompano Beach
20492	2018-10-26	Mary	Smith	CORPORATE	Tallahassee
20745	2018-10-29	Mary	Morrison	HOME_OFFICE	Brandon
20877	2018-10-31	Jose	Smith	CORPORATE	Miami
21239	2018-11-06	Patricia	Smith	CORPORATE	Fort Lauderdale
21278	2018-11-06	Mary	Harris	CORPORATE	Miami
22082	2018-11-18	Mary	Weaver	CONSUMER	Miami
22188	2018-11-19	Mary	Holmes	CONSUMER	Pompano Beach
22219	2018-11-20	James	Trevino	CONSUMER	Miami
22337	2018-11-22	Katherine	Roers	CONSUMER	Hollywood

Problem Statement #4

List all products that fall in one of the following categories: 'Computers', 'Toys', 'Tennis & Racquet'. Include the products' names, category, department, and price.

Query #4

```
select p.product_name, c.category_name, p.product_price, d.department_name from product p
inner join category c on c.category_id=p.category_id
inner join department d on d.department_id =p.department_id
where c.category_name in ('Computers', 'Toys', 'Tennis & Racquet');
```

```
18 -- ans4 Vishesh dabas - 18/09/2023
```

```
19 • select p.product_name, c.category_name, p.product_price, d.department_name from product p
```

```
20 inner join category c on c.category_id=p.category_id
```

```
21 inner join department d on d.department_id =p.department_id
```

```
22 where c.category_name in ('Computers', 'Toys', 'Tennis & Racquet');
```

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

product_name	category_name	product_price	department_name
Nike Mens Comfort 2 Slide	Tennis & Racquet	44.99	Fitness
Dell Laptop	Computers	1500.00	Technology
Toys	Toys	11.54	Fan Shop

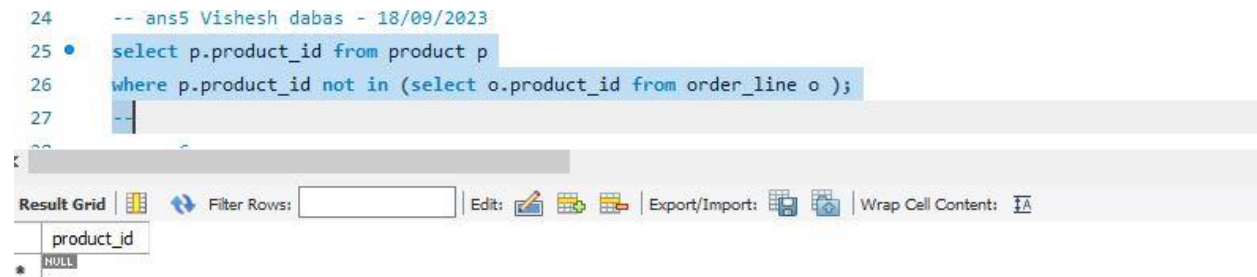
Problem Statement #5

TerpBuy is considering reducing its product offerings. Which products have not yet been sold? Include the name, category, and department for each such product.

Query #5

```
select p.product_id from product p
where p.product_id not in (select o.product_id from order_line o );
```

```
24 -- ans5 Vishesh dabas - 18/09/2023
25 • select p.product_id from product p
26 where p.product_id not in (select o.product_id from order_line o );
27 --
```



The screenshot shows a database query editor with a SQL query and a partial result grid. The query is: `select p.product_id from product p where p.product_id not in (select o.product_id from order_line o);`. The result grid shows a single column labeled `product_id` with a value of `NULL`.

product_id
NULL

Problem Statement #6

List the names of all cities from where orders are shipped. Also, for such cities, find the number of orders for which shipping was delayed. Sort the list of cities in order from the highest to the least number of shipping orders.

Query #6

```
SELECT orders.order_city, COUNT(*) AS 'Total Orders',  
COUNT(CASE WHEN actual_shipping_days - scheduled_shipping_days > 0 THEN 1 END)  
AS 'Number of Delayed Orders' FROM orders GROUP BY orders.order_city  
ORDER BY COUNT(CASE WHEN actual_shipping_days - scheduled_shipping_days > 0  
THEN 1 END) DESC;
```

```
51 -- ans6 Vishesh dabas - 18/09/2023  
52 • SELECT orders.order_city, COUNT(*) AS 'Total Orders',  
53 COUNT(CASE WHEN actual_shipping_days - scheduled_shipping_days > 0 THEN 1 END)  
54 AS 'Number of Delayed Orders' FROM orders GROUP BY orders.order_city  
55 ORDER BY COUNT(CASE WHEN actual_shipping_days - scheduled_shipping_days > 0 THEN 1 END) DESC;
```

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

	order_city	Total Orders	Number of Delayed Orders
▶	Bangalore	76	51
	Mumbai	69	45
	Pune	68	41
	Delhi	72	37
	Chennai	62	32
	Surat	46	31
	Visakhapatnam	42	30
	Hyderabad	46	29
	Gorakhpur	39	27
	Ajmer	39	26

Result 45 x

The <CASE WHEN> function helps us to generate a count of Delayed or Total orders to put in our Query when the query is grouped by the <order_city>, Then we can output that count in the final query.

Problem Statement #7

How many customers are there in each segment? Show the most popular segment at the top of the result. Incorporate a column alias in the result.

Query #7

```
select distinct segment, count(customer_id) as 'Most Popular Segment by count of customers'
from customer group by segment;
```

```
95  -- ans7 Vishesh dabas - 18/09/2023
96  select distinct segment, count(customer_id) as 'Most Popular Segment by count of customers'
97  from customer group by segment;
98  --
```

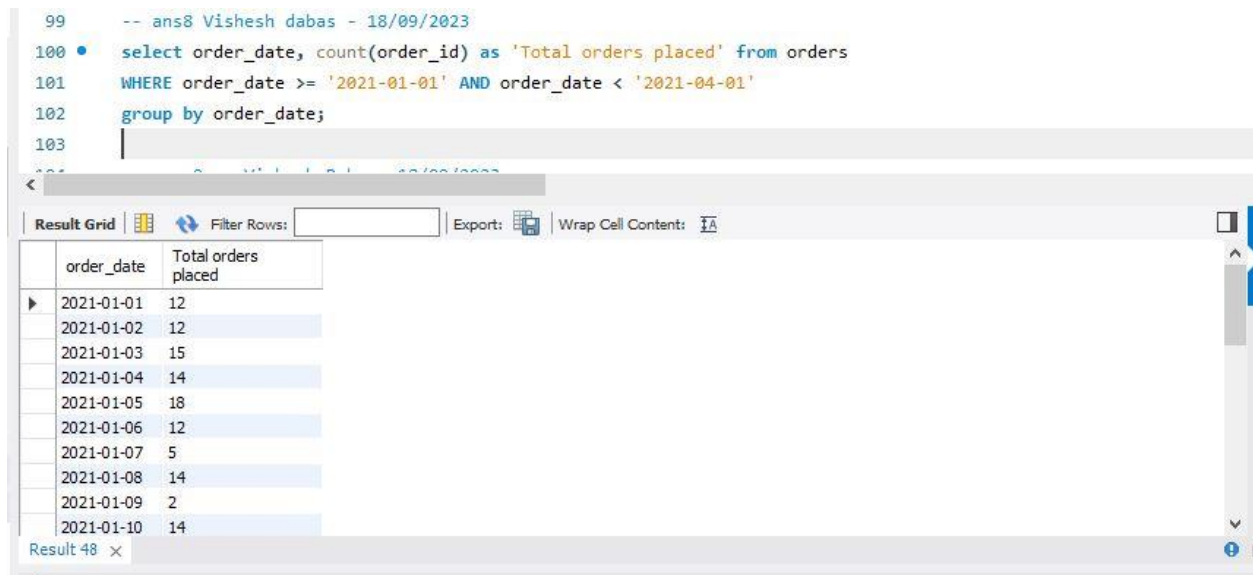
Result Grid		Filter Rows:	Export:	Wrap Cell Content:
segment	Most Popular Segment by count of customers			
CONSUMER	2312			
CORPORATE	1312			
HOME_OFFICE	837			

Problem Statement #8

How many orders were placed in the first quarter of 2021?

Query #8

```
select order_date, count(order_id) as 'Total orders placed' from orders
WHERE order_date >= '2021-01-01' AND order_date < '2021-04-01'
group by order_date;
```



The screenshot shows a SQL query execution interface. The query is as follows:

```
99 -- ans8 Vishesh dabas - 18/09/2023
100 • select order_date, count(order_id) as 'Total orders placed' from orders
101 WHERE order_date >= '2021-01-01' AND order_date < '2021-04-01'
102 group by order_date;
103
```

Below the query, the results are displayed in a table with the following columns: order_date and Total orders placed. The table contains 10 rows of data for the first 10 days of January 2021.

order_date	Total orders placed
2021-01-01	12
2021-01-02	12
2021-01-03	15
2021-01-04	14
2021-01-05	18
2021-01-06	12
2021-01-07	5
2021-01-08	14
2021-01-09	2
2021-01-10	14

The interface also includes a 'Result Grid' tab, a 'Filter Rows' input field, and an 'Export' button. The status bar at the bottom indicates 'Result 48'.

Problem Statement #9

List in alphabetical order all states supporting multiple customer segments.

Query #9

```
select count(distinct segment) as 'Number of segments supported', state from customer group by state;
```

```
103 -- ans 9 -- Vishesh Dabas, 18/09/2023
104 • select count(distinct segment) as 'Number of segments supported', state from customer group by state;
105
```

	Number of segments supported	state
▶	2	AR
	3	AZ
	3	CA
	3	CO
	3	CT
	3	DC
	3	DE
	3	FL
	3	GA
	3	HI
	3	...

Result 24 x

Problem Statement #10

To help the commercial sales department with its marketing, find all customers in the corporate segment who have not placed any orders. Include each customers' first name, last name, street, city, state, and zip code. Sort the results by the last name first and then by the first name.

Query #10

```
select c.first_name, c.last_name, c.city, c.state, c.zipcode from customer c
where c.customer_Id not in (select o.customer_Id from orders o ) and c.segment
='CORPORATE'
order by c.last_name, c.first_name ;
```

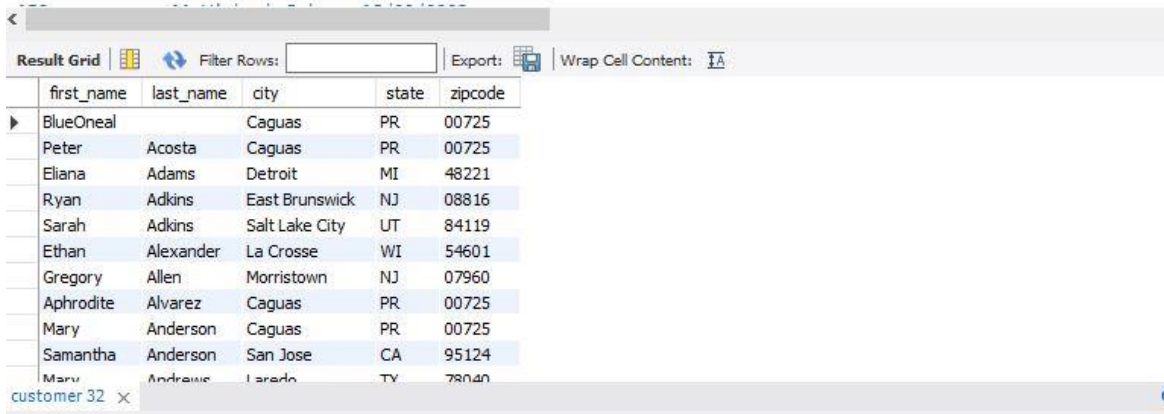
```
145 -- ans 10 Vishesh Dabas, 18/09/2023
```

```
146 • select c.first_name, c.last_name, c.city, c.state, c.zipcode from customer c
```

```
147 where c.customer_Id not in (select o.customer_Id from orders o ) and c.segment ='CORPORATE'
```

```
148 order by c.last_name, c.first_name ;
```

```
149
```



The screenshot shows a database query result grid. The grid has columns for first_name, last_name, city, state, and zipcode. The results are sorted by last_name and then by first_name. The first row is BlueOneal, and the last row is Marv. The grid also includes a Filter Rows section and an Export button.

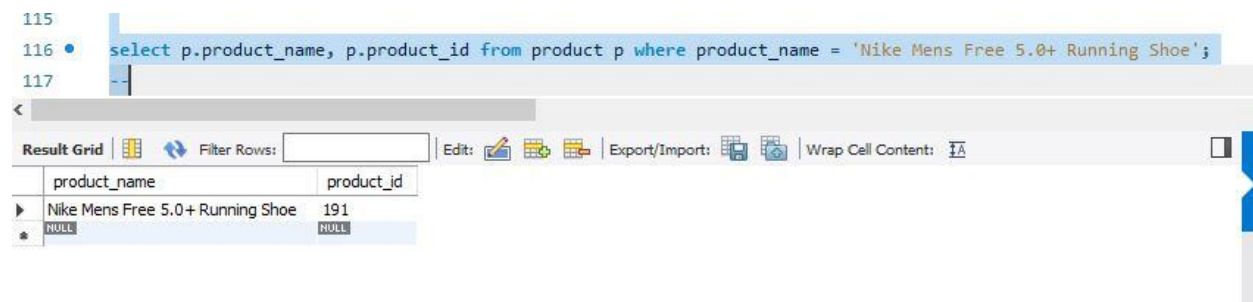
first_name	last_name	city	state	zipcode
BlueOneal		Caguas	PR	00725
Peter	Acosta	Caguas	PR	00725
Eliana	Adams	Detroit	MI	48221
Ryan	Adkins	East Brunswick	NJ	08816
Sarah	Adkins	Salt Lake City	UT	84119
Ethan	Alexander	La Crosse	WI	54601
Gregory	Allen	Morristown	NJ	07960
Aphrodite	Alvarez	Caguas	PR	00725
Mary	Anderson	Caguas	PR	00725
Samantha	Anderson	San Jose	CA	95124
Marv	Andrews	Las Vegas	NV	89101

Problem Statement #11

There has been a recall of the product Nike Mens Free 5.0+ Running Shoe. TerpBuy would have to offer a discount coupon to all customers who purchased this product. Find all orders that included this product as a part of the purchase. For all such orders, list the customers' first names, last names, street, state, zip code, and order date. Each customer can be offered only one discount coupon. Hence, do not list the same customer more than once.

Query #11

```
select p.product_name, p.product_id from product p where product_name = 'Nike Mens Free 5.0+ Running Shoe';
```



The screenshot shows a database query interface. The SQL query is: `select p.product_name, p.product_id from product p where product_name = 'Nike Mens Free 5.0+ Running Shoe';`. The results are displayed in a table with two columns: `product_name` and `product_id`. The first row shows 'Nike Mens Free 5.0+ Running Shoe' with a product_id of 191. The second row shows NULL values for both columns.

product_name	product_id
Nike Mens Free 5.0+ Running Shoe	191
NULL	NULL

Part 1 shows us the product Id of 'Nike Mens Free 5.0+ Running Shoe'; Which can be used for further analysis

Part 2

```
select c.first_name, c.last_name, c.street, c.state, c.zipcode, ol.order_id from customer c
inner join orders o on o.customer_Id = c.customer_Id inner join order_line ol on ol.order_id=
o.order_id
where product_id = 191 group by ol.order_id HAVING COUNT(*) = 1;
```

```

111 -- Nike Mens Free 5.0+ Running Shoe ans11 - Vishesh Dabas - 18/09/2023
112 • select c.first_name, c.last_name, c.street, c.state, c.zipcode, ol.order_id from customer c
113 inner join orders o on o.customer_Id = c.customer_Id inner join order_line ol on ol.order_id= o.order_id
114 where product_id = 191 group by ol.order_id HAVING COUNT(*) = 1; |
115

```

	first_name	last_name	street	state	zipcode	order_id
▶	Mary	Reynolds	4823 Broad Route	OR	97045	20338
	Mary	Smith	3385 Cotton Wharf	CA	95051	20364
	Wayne	Hardy	4132 Broad Gate Lane	TX	75150	20368
	Nicholas	Smith	603 Green Sky Promenade	LA	70072	20380
	Louis	Bishop	5192 Foggy Elk Village	PR	00725	20381
	Jonathan	Costa	849 Noble Apple Private	CA	91402	20454
	Mary	Smith	5340 Quaking Panda Forest	FL	32308	20492
	Mary	Lloyd	6035 Foggy Link	PR	00725	20548
	Justin	Smith	338 Heather Orchard	AZ	85029	20721
	Virginia	Sanders	1801 Jagged Dale Park	TX	78704	20758
	Robert	Smith	1987 Indian Autumn Swale	CO	80631	20828

Problem Statement #12

Premium customers are those customers who have placed orders with order amounts greater than the average order amount. For each customer, find the first and last names, and the order amount for all orders that exceeded the average order amount.

Query #12

```

SELECT c.first_name, c.last_name, ol.total_price
FROM customer c INNER JOIN orders o ON o.customer_Id = c.customer_Id
INNER JOIN order_line ol ON ol.order_id = o.order_id
where ol.total_price > (SELECT AVG(total_price) FROM order_line) ;

```

```

126 -- ans 12 Vishesh dabas - 18/09/2023
127 • SELECT c.first_name, c.last_name, ol.total_price
128 FROM customer c INNER JOIN orders o ON o.customer_Id = c.customer_Id
129 INNER JOIN order_line ol ON ol.order_id = o.order_id
130 where ol.total_price > (SELECT AVG(total_price) FROM order_line) ;

```

<
 Result Grid
 Filter Rows:
 Export:
 Wrap Cell Content:
 Fetch rows:

	first_name	last_name	total_price
▶	Phillip	Mcgee	399.98
	Mary	Reynolds	399.96
	Mary	Reynolds	239.96
	Mary	Smith	399.98
	Mary	Hill	399.98
	Mary	Hill	299.98
	Wayne	Hardy	499.95
	Nicholas	Smith	299.98
	Nicholas	Smith	299.97
	Louis	Bishop	399.98
	James	Smith	299.98

Result 38 ×