



E-COMMERCE (TARGET) SALES DATA ANALYSIS USING SQL

Prepared by: Younas Khan

LIST ALL UNIQUE CITIES WHERE CUSTOMERS ARE LOCATED.

-- List all unique cities where customers are located.

```
SELECT DISTINCT  
  customer_city  
FROM  
  customers;
```

	customer_city
▶	franca
	sao bernardo do campo
	sao paulo
	mogi das cruzeiras
	campinas
	jaraguá do sul
	timoteo
	curitiba
	belo horizonte
	montes claros
	rio de janeiro
	lencois paulista



COUNT THE NUMBER OF ORDERS PLACED IN 2017.

```
-- Count the number of orders placed in 2017.
```

```
SELECT
```

```
    COUNT(order_id)
```

```
FROM
```

```
    orders
```

```
WHERE
```

```
    YEAR(order_purchase_timestamp) = 2017
```



	count(order_id)
▶	45101

FIND THE TOTAL SALES PER CATEGORY.

-- Find the total sales per category.

• SELECT

products.product_category category,
ROUND(SUM(payment_value), 2) sales

FROM

products

JOIN

order_items ON products.product_id = order_items.product_id

JOIN

payments ON payments.order_id = order_items.order_id

GROUP BY category

	category	sales
▶	perfumery	506738.66
	Furniture Decoration	1430176.39
	telephony	486882.05
	bed table bath	1712553.67
	automotive	852294.33
	computer accessories	1585330.45
	housewares	1094758.13
	babies	539845.66
	toys	619037.69
	Furniture office	646826.49

CALCULATE THE PERCENTAGE OF ORDERS THAT WERE PAID IN INSTALLMENTS.

```
-- Calculate the percentage of orders that were paid in installments.  
  
SELECT  
    (SUM(CASE  
        WHEN payment_installments >= 1 THEN 1  
        ELSE 0  
    END)) / COUNT(*) * 100 AS Percentage_orders_paid_installmensts  
FROM  
    payments
```



	Percentage_orders_paid_installmensts
▶	99.9981

- **COUNT THE NUMBER OF CUSTOMERS FROM EACH STATE.**

```
-- Count the number of customers from each state.
```

```
SELECT
```

```
    customer_state, COUNT(customer_id) as number_customers
```

```
FROM
```

```
    customers
```

```
GROUP BY customer_state
```

	customer_state	number_customers
▶	SP	41746
	SC	3637
	MG	11635
	PR	5045
	RJ	12852
	RS	5466
	PA	975
	GO	2020
	ES	2033

CALCULATE THE NUMBER OF ORDERS PER MONTH IN 2018.

```
-- Calculate the number of orders per month in 2018.
```

```
SELECT
```

```
    MONTHname(order_purchase_timestamp) months,
```

```
    COUNT(order_id) order_count
```

```
FROM
```

```
    orders
```

```
WHERE
```

```
    YEAR(order_purchase_timestamp) = 2018
```

```
GROUP BY months
```



	months	order_count
▶	July	6292
	August	6512
	February	6728
	June	6167
	March	7211
	January	7269
	May	6873
	April	6939

FIND THE AVERAGE NUMBER OF PRODUCTS PER ORDER, GROUPED BY CUSTOMER CITY.

```
-- Find the average number of products per order, grouped by customer city.

with count_per_order as
(select orders.order_id, orders.customer_id, count(order_items.order_id) as ordercount
 from orders join order_items
 on orders.order_id = order_items.order_id
 group by orders.order_id, orders.customer_id)

SELECT
  customers.customer_city,
  ROUND(AVG(count_per_order.ordercount), 2) average_orders
FROM
  customers
  JOIN
  count_per_order ON customers.customer_id = count_per_order.customer_id
GROUP BY customers.customer_city;
```

	customer_city	average_orders
▶	sao paulo	1.16
	sao jose dos campos	1.14
	porto alegre	1.17
	indaial	1.12
	treze tilias	1.27
	rio de janeiro	1.15
	mario campos	1.33

CALCULATE THE PERCENTAGE OF TOTAL REVENUE CONTRIBUTED BY EACH PRODUCT CATEGORY.

-- Calculate the percentage of total revenue contributed by each product category.

```
SELECT
    products.product_category category,
    round((SUM(payments.payment_value) / (select sum(payment_value) from payments))*100,2) sales_percentage
FROM
    products
    JOIN
    order_items ON products.product_id = order_items.product_id
    JOIN
    payments ON payments.order_id = order_items.order_id
GROUP BY category order by sales_percentage desc;
```

	category	sales_percentage
▶	bed table bath	10.7
	HEALTH BEAUTY	10.35
	computer accessories	9.9
	Furniture Decoration	8.93
	Watches present	8.93
	sport leisure	8.7
	housewares	6.84

IDENTIFY THE CORRELATION BETWEEN PRODUCT PRICE AND THE NUMBER OF TIMES A PRODUCT HAS BEEN PURCHASED.

-- Identify the correlation between product price and the number of times a product has been purchased.

SELECT

products.product_category,
COUNT(order_items.product_id) AS time_order,
ROUND(AVG(order_items.price), 2) AS price

FROM

products

JOIN

order_items ON products.product_id = order_items.product_id

GROUP BY products.product_category;

	product_category	time_order	price
▶	HEALTH BEAUTY	9670	130.16
	sport leisure	8641	114.34
	Cool Stuff	3796	167.36
	computer accessories	7827	116.51
	Watches present	5991	201.14
	housewares	6964	90.79
	electronics	2767	57.91
	HULL	1603	112
	toys	4117	117.55

CALCULATE THE TOTAL REVENUE GENERATED BY EACH SELLER, AND RANK THEM BY REVENUE.

-- Calculate the total revenue generated by each seller, and rank them by revenue.

```
select *, dense_rank() over(order by revenue desc) as rank_revenue from
(SELECT
    order_items.seller_id, SUM(payments.payment_value) as revenue
FROM
    order_items
    JOIN
    payments ON order_items.order_id = payments.order_id
GROUP BY order_items.seller_id) as a
```

	seller_id	revenue	rank_revenue
▶	7c67e1448b00f6e969d365cea6b010ab	507166.9073021412	1
	1025f0e2d44d7041d6cf58b6550e0bfa	308222.0398402214	2
	4a3ca9315b744ce9f8e9374361493884	301245.26976528764	3
	1f50f920176fa81dab994f9023523100	290253.42012761533	4
	53243585a1d6dc2643021fd1853d8905	284903.0804977417	5
	da8622b14eb17ae2831f4ac5b9dab84a	272219.31931465864	6
	4869f7a5dfa277a7dca6462dcf3b52b2	264166.1209387779	7
	955fee9216a65b617aa5c0531780ce60	236322.30050226487	8

CALCULATE THE MOVING AVERAGE OF ORDER VALUES FOR EACH CUSTOMER OVER THEIR ORDER HISTORY.

● Calculate the moving average of order values for each customer over their order history.

```
SELECT a.customer_id,  
       a.order_purchase_timestamp,  
       a.payment_value, -- This column will show the payment for each order  
       AVG(a.payment_value) OVER (PARTITION BY a.customer_id  
                                   ORDER BY a.order_purchase_timestamp  
                                   ROWS BETWEEN 2 PRECEDING AND CURRENT ROW  
       ) AS mov_avg -- This calculates the moving average for each customer FROM  
       (SELECT orders.customer_id,  
              orders.order_purchase_timestamp,  
              payments.payment_value  
       FROM payments JOIN orders  
       ON payments.order_id = orders.order_id) AS a;
```

customer_id	order_purchase_timestamp	payment_value	mov_avg
08c5351a6aca1c1589a38f244edeee9d	2016-09-04 21:15:19	136.23	136.22999572753906
683c54fc24d40ee9f8a6fc179fd9856c	2016-09-05 00:15:34	75.06	75.05999755859375
622e13439d6b5a0b486c435618b2679e	2016-09-13 15:24:19	40.95	40.95000076293945
b106b360fe2ef8849fbbd056f777b4d5	2016-10-02 22:07:52	109.34	109.33999633789062
355077684019f7f60a031656bd7262b8	2016-10-03 09:44:50	45.46	45.459999084472656
7ec40b22510fdbea1b08921dd39e63d8	2016-10-03 16:56:50	39.09	39.09000015258789
70fc57eeae292675927697fe03ad3ff5	2016-10-03 21:01:41	35.61	35.61000061035156
6f989332712d3222b6571b1cf5b835ce	2016-10-03 21:13:36	53.73	53.72999954223633

CALCULATE THE YEAR-OVER-YEAR GROWTH RATE OF TOTAL SALES.

-- Calculate the cumulative sales per month for each year.

```
select years,months, payment, sum(payment)
over(order by years, months) comulative_sales
from
(select year(orders.order_purchase_timestamp) as years,
month(orders.order_purchase_timestamp) as months,
round(sum(payments.payment_value),2) as payment
from orders
join payments
on orders.order_id= payments.order_id
group by years,months order by years, months) as a;
```

	years	months	payment	comulative_sales
	2016	9	252.24	252.24
	2016	10	59090.48	59342.72
	2016	12	19.62	59362.34000000000004
	2017	1	138488.04	197850.38
	2017	2	291908.01	489758.39
	2017	3	449863.6	939621.99
	2017	4	417788.03	1357410.02
	2017	5	592918.82	1950328.8399999999

CALCULATE THE YEAR-OVER-YEAR GROWTH RATE OF TOTAL SALES.

```
-- Calculate the year-over-year growth rate of total sales
```

```
SELECT
```

```
    YEAR(orders.order_purchase_timestamp) AS years,
```

```
    ROUND(SUM(payments.payment_value), 2) AS payment
```

```
FROM
```


```
    orders
```

```
    JOIN
```

```
    payments ON orders.order_id = payments.order_id
```

```
GROUP BY years
```

```
ORDER BY years
```



	years	payment
▶	2016	59362.34
	2017	7249746.73
	2018	8699763.05

IDENTIFY THE TOP 3 CUSTOMERS WHO SPENT THE MOST MONEY IN EACH YEAR.

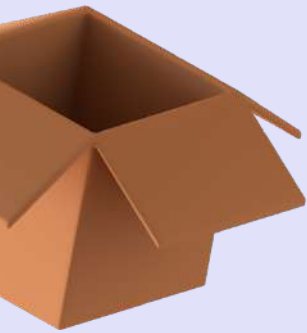

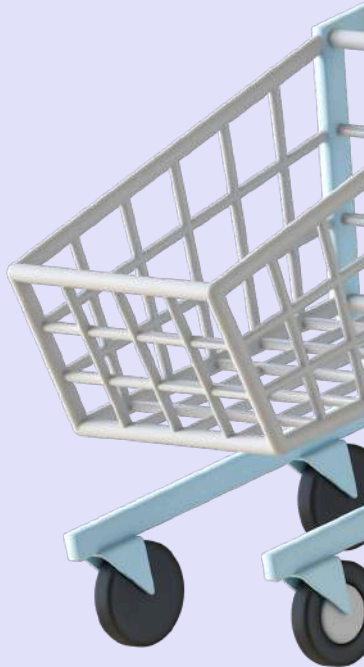
- Identify the top 3 customers who spent the most money in each year

```
● select year, customer_id, payment, ranking
from
(select year(orders.order_purchase_timestamp) as year,
orders.customer_id,
sum(payments.payment_value) payment,
dense_rank() over (partition by year(orders.order_purchase_timestamp)
order by sum(payments.payment_value) desc) as ranking
from orders
join payments on orders.order_id = payments.order_id
group by
year(orders.order_purchase_timestamp), orders.customer_id) as a
where ranking <= 3;
```

	year	customer_id	payment	ranking
	2016	1d34ed25963d5aae4cf3d7f3a4cda173	1400.739990234375	2
	2016	4a06381959b6670756de02e07b83815f	1227.780029296875	3
	2017	1617b1357756262bfa56ab541c47bc16	13664.080078125	1
	2017	c6e2731c5b391845f6800c97401a43a9	6929.31005859375	2
	2017	3fd6777bbce08a352fddd04e4a7cc8f6	6726.66015625	3
	2018	ec5b2ba62e574342386871631fafd3fc	7274.8798828125	1
	2018	f48d464a0baaea338cb25f816991ab1f	6922.2099609375	2
	2018	e0a2412720e9ea4f26c1ac985f6a7358	4809.43994140625	3



THANK YOU

- **FOR TAKING THE TIME TO REVIEW MY PROJECT ON E-COMMERCE SALES DATA ANALYSIS. I HOPE THIS PROJECT PROVIDED VALUABLE INSIGHTS INTO DATA-DRIVEN DECISION-MAKING IN THE E-COMMERCE INDUSTRY.**
 - **IF YOU HAVE ANY QUESTIONS OR FEEDBACK, FEEL FREE TO REACH OUT!**
- 
- 
- 
- 