SQL - Capstone Project

- **AIM:** The major aim of this project is to gain insight into the sales data of Amazon to understand the different factors that affect sales of the different branches.
- **About Data:** This dataset contains sales transactions from three different branches of Amazon, respectively located in Mandalay, Yangon and Naypyitaw. The data contains 17 columns and 1000 rows.

AmazonSQL*

Limit to 1000 rows

```
use capstone_sql_amazon;
select * from amazon;
```

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

invoice_id	branch	city	customer_type	gender	product_line	unit_price	quantity	VAT	total	date	time	payment	cogs	gross_margin_percentage	gross_income	rating
750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	2019-01-05	13:08:00	Ewallet	522.83	4.761904762	26.1415	9.1
226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.82	80.22	2019-03-08	10:29:00	Cash	76.4	4.761904762	3.82	9.6
631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	2019-03-03	13:23:00	Credit card	324.31	4.761904762	16.2155	7.4
123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.288	489.048	2019-01-27	20:33:00	Ewallet	465.76	4.761904762	23.288	8.4
373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2019-02-08	10:37:00	Ewallet	604.17	4.761904762	30.2085	5.3
699-14-3026	C	Naypyitaw	Normal	Male	Electronic accessories	85.39	7	29.8865	627.6165	2019-03-25	18:30:00	Ewallet	597.73	4.761904762	29.8865	4.1
355-53-5943	A	Yangon	Member	Female	Electronic accessories	68.84	6	20.652	433.692	2019-02-25	14:36:00	Ewallet	413.04	4.761904762	20.652	5.8
315-22-5665	C	Naypyitaw	Normal	Female	Home and lifestyle	73.56	10	36.78	772.38	2019-02-24	11:38:00	Ewallet	735.6	4.761904762	36.78	8
665-32-9167	A	Yangon	Member	Female	Health and beauty	36.26	2	3.626	76.146	2019-01-10	17:15:00	Credit card	72.52	4.761904762	3.626	7.2
692-92-5582	B	Mandalay	Member	Female	Food and beverages	54.84	3	8.226	172.746	2019-02-20	13:27:00	Credit card	164.52	4.761904762	8.226	5.9
351-62-0822	B	Mandalay	Member	Female	Fashion accessories	14.48	4	2.896	60.816	2019-02-06	18:07:00	Ewallet	57.92	4.761904762	2.896	4.5
529-56-3974	B	Mandalay	Member	Male	Electronic accessories	25.51	4	5.102	107.142	2019-03-09	17:03:00	Cash	102.04	4.761904762	5.102	6.8
365-64-0515	A	Yangon	Normal	Female	Electronic accessories	46.95	5	11.7375	246.4875	2019-02-12	10:25:00	Ewallet	234.75	4.761904762	11.7375	7.1
252-56-2699	A	Yangon	Normal	Male	Food and beverages	43.19	10	21.595	453.495	2019-02-07	16:48:00	Ewallet	431.9	4.761904762	21.595	8.2
829-34-3910	A	Yangon	Normal	Female	Health and beauty	71.38	10	35.69	749.49	2019-03-29	19:21:00	Cash	713.8	4.761904762	35.69	5.7
299-46-1805	B	Mandalay	Member	Female	Sports and travel	93.72	6	28.116	590.436	2019-01-15	16:19:00	Cash	562.32	4.761904762	28.116	4.5
656-95-9349	A	Yangon	Member	Female	Health and beauty	68.93	7	24.1255	506.6355	2019-03-11	11:03:00	Credit card	482.51	4.761904762	24.1255	4.6
765-26-6951	A	Yangon	Normal	Male	Sports and travel	72.61	6	21.783	457.443	2019-01-01	10:39:00	Credit card	435.66	4.761904762	21.783	6.9
329-62-1586	A	Yangon	Normal	Male	Food and beverages	54.67	3	8.2005	172.2105	2019-01-21	18:00:00	Credit card	164.01	4.761904762	8.2005	8.6
319-50-3348	B	Mandalay	Normal	Female	Home and lifestyle	40.3	2	4.03	84.63	2019-03-11	15:30:00	Ewallet	80.6	4.761904762	4.03	4.4

amazon 1

- **Feature Engineering:** This will help us generate some new columns from existing ones.

- Add a new column named timeofday to give insight of sales in the Morning, Afternoon and Evening. This will help answer the question on which part of the day most sales are made.

AmazonSQL*

Limit to 1000 rows

```sql
use capstone_sql_amazon;
select * from amazon;

ALTER TABLE amazon ADD COLUMN timeofday VARCHAR(10);

UPDATE amazon
SET timeofday =
CASE WHEN CAST(SUBSTRING_INDEX(time, ':', 1) AS UNSIGNED) >= 0 AND CAST(SUBSTRING_INDEX(time, ':', 1) AS UNSIGNED) < 12 THEN 'Morning'
WHEN CAST(SUBSTRING_INDEX(time, ':', 1) AS UNSIGNED) >= 12 AND CAST(SUBSTRING_INDEX(time, ':', 1) AS UNSIGNED) < 18 THEN 'Afternoon'
ELSE 'Evening' END;
```

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

branch	city	customer_type	gender	product_line	unit_price	quantity	VAT	total	date	time	payment	cogs	gross_margin_percentage	gross_income	rating	timeofday
A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	2019-01-05	13:08:00	Ewallet	522.83	4.761904762	26.1415	9.1	Afternoon
C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.82	80.22	2019-03-08	10:29:00	Cash	76.4	4.761904762	3.82	9.6	Morning
A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	2019-03-03	13:23:00	Credit card	324.31	4.761904762	16.2155	7.4	Afternoon
A	Yangon	Member	Male	Health and beauty	58.22	8	23.288	489.048	2019-01-27	20:33:00	Ewallet	465.76	4.761904762	23.288	8.4	Evening
A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2019-02-08	10:37:00	Ewallet	604.17	4.761904762	30.2085	5.3	Morning
C	Naypyitaw	Normal	Male	Electronic accessories	85.39	7	29.8865	627.6165	2019-03-25	18:30:00	Ewallet	597.73	4.761904762	29.8865	4.1	Evening
A	Yangon	Member	Female	Electronic accessories	68.84	6	20.652	433.692	2019-02-25	14:36:00	Ewallet	413.04	4.761904762	20.652	5.8	Afternoon
C	Naypyitaw	Normal	Female	Home and lifestyle	73.56	10	36.78	772.38	2019-02-24	11:38:00	Ewallet	735.6	4.761904762	36.78	8	Morning
A	Yangon	Member	Female	Health and beauty	36.26	2	3.626	76.146	2019-01-10	17:15:00	Credit card	72.52	4.761904762	3.626	7.2	Afternoon
B	Mandalay	Member	Female	Food and beverages	54.84	3	8.226	172.746	2019-02-20	13:27:00	Credit card	164.52	4.761904762	8.226	5.9	Afternoon
B	Mandalay	Member	Female	Fashion accessories	14.48	4	2.896	60.816	2019-02-06	18:07:00	Ewallet	57.92	4.761904762	2.896	4.5	Evening
B	Mandalay	Member	Male	Electronic accessories	25.51	4	5.102	107.142	2019-03-09	17:03:00	Cash	102.04	4.761904762	5.102	6.8	Afternoon

amazon 5

-

- Add a new column named dayname that contains the extracted days of the week on which the given transaction took place (Mon, Tue, Wed, Thur, Fri). This will help answer the question on which week of the day each branch is busiest.

AmazonSQL*

Limit to 1000 rows

```sql
ALTER TABLE amazon ADD COLUMN dayname VARCHAR(3);

UPDATE amazon
SET dayname =
    CASE
        WHEN DAYOFWEEK(date) = 2 THEN 'Mon'
        WHEN DAYOFWEEK(date) = 3 THEN 'Tue'
        WHEN DAYOFWEEK(date) = 4 THEN 'Wed'
        WHEN DAYOFWEEK(date) = 5 THEN 'Thu'
        WHEN DAYOFWEEK(date) = 6 THEN 'Fri'
    END;
```

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

city	customer_type	gender	product_line	unit_price	quantity	VAT	total	date	time	payment	cogs	gross_margin_percentage	gross_income	rating	timeofday	dayname
Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	2019-01-05	13:08:00	Ewallet	522.83	4.761904762	26.1415	9.1	Afternoon	NULL
Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.82	80.22	2019-03-08	10:29:00	Cash	76.4	4.761904762	3.82	9.6	Morning	Fri
Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	2019-03-03	13:23:00	Credit card	324.31	4.761904762	16.2155	7.4	Afternoon	NULL
Yangon	Member	Male	Health and beauty	58.22	8	23.288	489.048	2019-01-27	20:33:00	Ewallet	465.76	4.761904762	23.288	8.4	Evening	NULL
Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2019-02-08	10:37:00	Ewallet	604.17	4.761904762	30.2085	5.3	Morning	Fri
Naypyitaw	Normal	Male	Electronic accessories	85.39	7	29.8865	627.6165	2019-03-25	18:30:00	Ewallet	597.73	4.761904762	29.8865	4.1	Evening	Mon
Yangon	Member	Female	Electronic accessories	68.84	6	20.652	433.692	2019-02-25	14:36:00	Ewallet	413.04	4.761904762	20.652	5.8	Afternoon	Mon
Naypyitaw	Normal	Female	Home and lifestyle	73.56	10	36.78	772.38	2019-02-24	11:38:00	Ewallet	735.6	4.761904762	36.78	8	Morning	NULL
Yangon	Member	Female	Health and beauty	36.26	2	3.626	76.146	2019-01-10	17:15:00	Credit card	72.52	4.761904762	3.626	7.2	Afternoon	Thu
Mandalay	Member	Female	Food and beverages	54.84	3	8.226	172.746	2019-02-20	13:27:00	Credit card	164.52	4.761904762	8.226	5.9	Afternoon	Wed
Mandalay	Member	Female	Fashion accessories	14.48	4	2.896	60.816	2019-02-06	18:07:00	Ewallet	57.92	4.761904762	2.896	4.5	Evening	Wed
Mandalay	Member	Male	Electronic accessories	25.51	4	5.102	107.142	2019-03-09	17:03:00	Cash	102.04	4.761904762	5.102	6.8	Afternoon	NULL

-

- Add a new column named monthname that contains the extracted months of the year on which the given transaction took place (Jan, Feb, Mar). Help determine which month of the year has the most sales and profit.

AmazonSQL*

Limit to 1000 rows

```sql
ALTER TABLE amazon ADD COLUMN monthname VARCHAR(3);

UPDATE amazon
SET monthname =
    CASE
        WHEN MONTH(date) = 1 THEN 'Jan'
        WHEN MONTH(date) = 2 THEN 'Feb'
        WHEN MONTH(date) = 3 THEN 'Mar'
        WHEN MONTH(date) = 4 THEN 'Apr'
        WHEN MONTH(date) = 5 THEN 'May'
        WHEN MONTH(date) = 6 THEN 'Jun'
        WHEN MONTH(date) = 7 THEN 'Jul'
        WHEN MONTH(date) = 8 THEN 'Aug'
        WHEN MONTH(date) = 9 THEN 'Sep'
        WHEN MONTH(date) = 10 THEN 'Oct'
        WHEN MONTH(date) = 11 THEN 'Nov'
        ELSE 'Dec'
    END;
```

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

customer_type	gender	product_line	unit_price	quantity	VAT	total	date	time	payment	cogs	gross_margin_percentage	gross_income	rating	timeofday	dayname	monthname
lember	Female	Health and beauty	74.69	7	26.1415	548.9715	2019-01-05	13:08:00	Ewallet	522.83	4.761904762	26.1415	9.1	Afternoon	NULL	Jan
ormal	Female	Electronic accessories	15.28	5	3.82	80.22	2019-03-08	10:29:00	Cash	76.4	4.761904762	3.82	9.6	Morning	Fri	Mar
ormal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	2019-03-03	13:23:00	Credit card	324.31	4.761904762	16.2155	7.4	Afternoon	NULL	Mar
lember	Male	Health and beauty	58.22	8	23.288	489.048	2019-01-27	20:33:00	Ewallet	465.76	4.761904762	23.288	8.4	Evening	NULL	Jan
ormal	Male	Sports and travel	86.31	7	30.2085	634.3785	2019-02-08	10:37:00	Ewallet	604.17	4.761904762	30.2085	5.3	Morning	Fri	Feb
ormal	Male	Electronic accessories	85.39	7	29.8865	627.6165	2019-03-25	18:30:00	Ewallet	597.73	4.761904762	29.8865	4.1	Evening	Mon	Mar
lember	Female	Electronic accessories	68.84	6	20.652	433.692	2019-02-25	14:36:00	Ewallet	413.04	4.761904762	20.652	5.8	Afternoon	Mon	Feb
ormal	Female	Home and lifestyle	73.56	10	36.78	772.38	2019-02-24	11:38:00	Ewallet	735.6	4.761904762	36.78	8	Morning	NULL	Feb
lember	Female	Health and beauty	36.26	2	3.626	76.146	2019-01-10	17:15:00	Credit card	72.52	4.761904762	3.626	7.2	Afternoon	Thu	Jan

amazon 7

Output

- ## Business Questions To Answer:

```sql
-- 1. What is the count of distinct cities in the dataset?
select city, count(city) count_cities from amazon group by city;

/*This SQL query counts the number of occurrences of each city in the "amazon" dataset and groups the results by city.
The count(city) function is used to count the occurrences of each city, and the result is stored in a column named count_cities.
The group by city statement groups the data by city so that the count is calculated for each unique city in the dataset.*/
```

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

city	count_cities
Yangon	340
Naypyitaw	328
Mandalay	332

-

```sql
-- 2. For each branch, what is the corresponding city?
select branch, city from amazon group by branch, city;

/*the query is asking for the city corresponding to each branch in the "amazon" dataset. It groups the data so that it shows the
relationship between each branch and the city where it is located.*/
```

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

branch	city
A	Yangon
C	Naypyitaw
B	Mandalay

-

```sql
-- 3. What is the count of distinct product lines in the dataset?
select product_line, count(product_line) count_productline from amazon group by product_line;

/*the query is asking how many times each product line appears in the "amazon" dataset. It then provides a count for each product line,
showing the number of times each product line is mentioned in the dataset.*/
```

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

product_line	count_productline
Health and beauty	152
Electronic accessories	170
Home and lifestyle	160
Sports and travel	166
Food and beverages	174
Fashion accessories	178

- AmazonSQL* | Limit to 1000 rows

```sql
-- 4. Which payment method occurs most frequently?
select max(payment) most_payment, count(*) payment_count from amazon group by payment order by payment_count desc limit 1;

/*This SQL query calculates the most frequently occurring payment method in the "amazon" dataset. It groups the data by the payment method and counts
the occurrences of each payment method using the count(*) function. The order by payment_count desc statement sorts the results in descending order
based on the payment count. The limit 1 clause ensures that only the first row of the sorted result set is returned, which corresponds to the payment
method that occurs most frequently.*/
```

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

most_payment	payment_count
Ewallet	345

-

```sql
-- 5. Which product line has the highest sales?
select product_line, sum(total) sales from amazon group by product_line order by sales desc limit 1;

/*This SQL query calculates the product line with the highest sales in the "amazon" dataset. It groups the data by product line and calculates the total sales
for each product line using the sum(total) function. The order by sales desc statement sorts the results in descending order based on the total sales.
The limit 1 clause ensures that only the first row of the sorted result set is returned, which corresponds to the product line with the highest sales.*/
```

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

product_line	sales
Food and beverages	56144.844000000005

```sql
-- 6. How much revenue is generated each month?
select month(STR_TO_DATE(date, '%Y-%m-%d')) as month, sum(gross_income) revenue from amazon group by month;

/*
This SQL query calculates the revenue generated each month in the "amazon" dataset. It uses the STR_TO_DATE(date, '%Y-%m-%d') function to convert the date
column into a date format, extracts the month from the date using the month() function, and groups the data by month. The sum(gross_income) function calculates
the total revenue for each month.*/
```

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

month	revenue
1	5537.707999999999
3	5212.167000000006
2	4629.494000000001

```sql
-- 7. In which month did the cost of goods sold reach its peak?
select month(STR_TO_DATE(date, '%Y-%m-%d')) as month, sum(total) revenue from amazon group by month order by revenue desc limit 1;

/*This SQL query calculates the month when the cost of goods sold (COGS) reached its peak in the "amazon" dataset. It converts the date column into a date format,
extracts the month from the date, and groups the data by month. The sum(total) function calculates the total revenue for each month.
The order by revenue desc statement sorts the results in descending order based on the total revenue. */
```

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

month	revenue
1	116291.86800000005

```sql
-- 8. Which product line generated the highest revenue?
select product_line, sum(unit_price * quantity) revenue from amazon group by product_line order by revenue desc limit 1;

/*the query is asking which product line generated the highest total revenue in the "amazon" dataset, and it provides the total revenue for that product line.*/
```

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

product_line	revenue
Food and beverages	53471.28000000006

```sql
-- 9. In which city was the highest revenue recorded?
select city, sum(gross_income) as total_revenue from amazon group by city order by total_revenue desc limit 1;

/*
This query finds the city where the highest total revenue was recorded in the dataset. It calculates the total revenue for each
city by summing the gross_income column, groups the results by city, and then orders them in descending order based on the total revenue.
The LIMIT 1 clause ensures that only the city with the highest total revenue is returned.*/
```

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

city	total_revenue
Naypyitaw	5265.176500000002

```sql
-- 10. Which product line incurred the highest Value Added Tax?
select product_line, sum(vat) as total_vat from amazon group by product_line order by total_vat desc limit 1;

/*
This query identifies the product line that incurred the highest Value Added Tax (VAT). It calculates the total VAT for each product line by
summing the vat column, groups the results by product line, and then orders them in descending order based on the total VAT. The LIMIT 1 clause ensures
that only the product line with the highest total VAT is returned.*/
```

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

product_line	total_vat
Food and beverages	2673.5639999999994

AmazonSQL*

Limit to 1000 rows

```sql
-- 11. For each product line, add a column indicating "Good" if its sales are above average, otherwise "Bad."
with avg_sales as (select avg(total) as avg_total from amazon)
select product_line, total, case when total > (select avg_total from avg_sales) then 'Good' else 'Bad' end as sales_indicator from amazon;

/*This query adds a column to each row in the dataset indicating whether the sales for that product line are above or below the average sales
across all product lines. It first calculates the average total sales (avg_total) using a subquery and assigns it to a temporary table avg_sales.
Then, for each row in the dataset, it compares the total sales (total) for that product line with the average total sales. If the total sales are
greater than the average, it assigns the value 'Good' to the sales_indicator column; otherwise, it assigns 'Bad'.*/
```

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

product_line	total	sales_indicator
Health and beauty	548.9715	Good
Electronic accessories	80.22	Bad
Home and lifestyle	340.5255	Good
Health and beauty	489.048	Good
Sports and travel	634.3785	Good
Electronic accessories	627.6165	Good
Electronic accessories	433.692	Good
Home and lifestyle	772.38	Good
Health and beauty	76.146	Bad
Food and beverages	172.746	Bad
Fashion accessories	60.816	Bad
Electronic accessories	107.142	Bad
Electronic accessories	246.4875	Bad
Food and beverages	453.495	Good
Health and beauty	749.49	Good
Sports and travel	590.436	Good
Health and beauty	506.6355	Good
Sports and travel	457.443	Good
Food and beverages	172.2105	Bad
Home and lifestyle	84.63	Bad

Result 18

- AmazonSQL*

Limit to 1000 rows

```sql
-- 12. Identify the branch that exceeded the average number of products sold.
with avg_products as (select avg(quantity) as avg_quantity from amazon)
select branch, sum(quantity) as total_quantity from amazon group by branch
having sum(quantity) > (select avg_quantity from avg_products);

/*This query identifies the branch that exceeded the average number of products sold. It first calculates the average quantity of products sold (avg_quantity)
using a subquery and assigns it to a temporary table avg_products. Then, it calculates the total quantity of products sold for each branch using the SUM(quantity)
function and groups the results by branch. Finally, it uses the HAVING clause to filter the results to only include branches where the total quantity of products
sold exceeds the average quantity.*/
```

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

branch	total_quantity
A	1859
C	1831
B	1820

```sql
-- 13. Which product line is most frequently associated with each gender?
select gender, product_line, count(*) as frequency from amazon group by gender, product_line order by gender, frequency desc;

/*This query determines which product line is most frequently associated with each gender in the dataset. It counts the occurrences of each combination
of gender and product line using the COUNT(*) function, groups the results by gender and product line, and then orders them first by gender and then by
frequency in descending order.*/
```

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

gender	product_line	frequency
Female	Fashion accessories	96
Female	Food and beverages	90
Female	Sports and travel	88
Female	Electronic accessories	84
Female	Home and lifestyle	79
Female	Health and beauty	64
Male	Health and beauty	88
Male	Electronic accessories	86
Male	Food and beverages	84
Male	Fashion accessories	82
Male	Home and lifestyle	81
Male	Sports and travel	78

Limit to 1000 rows

```sql
-- 14. Calculate the average rating for each product line.
select product_line, avg(rating) as average_rating from amazon group by product_line;

/*This query calculates the average rating for each product line in the dataset. It uses the AVG(rating) function to calculate the average rating
for each product line, grouping the results by product line. The result set includes the product line and its corresponding average rating.*/
```

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

product_line	average_rating
Health and beauty	7.003289473684212
Electronic accessories	6.92470588235294
Home and lifestyle	6.8375
Sports and travel	6.916265060240964
Food and beverages	7.113218390804598
Fashion accessories	7.029213483146067

```sql
-- 15. Count the sales occurrences for each time of day on every weekday.
select dayofweek(date) as weekday, dayname(date) as weekday_name, hour(time) as hour_of_day, count(*) as sales_occurrences
from amazon where dayofweek(date) between 2 and 6 group by dayofweek(date), hour(time) order by dayofweek(date), hour(time);

/*This query counts the sales occurrences for each hour of the day from Monday to Friday. It groups the data by weekday, weekday name, and hour of the day,
providing the count of sales occurrences for each combination.*/
```

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

weekday	weekday_name	hour_of_day	sales_occurrences
2	Monday	10	12
2	Monday	11	9
2	Monday	12	14
2	Monday	13	10
2	Monday	14	6
2	Monday	15	18
2	Monday	16	16
2	Monday	17	11
2	Monday	18	11
2	Monday	19	8
2	Monday	20	10
3	Tuesday	10	16
3	Tuesday	11	20
3	Tuesday	12	11
3	Tuesday	13	13
3	Tuesday	14	15
3	Tuesday	15	14
3	Tuesday	16	9
3	Tuesday	17	9
3	Tuesday	18	12

Result 22

```sql
-- 16. Identify the customer type contributing the highest revenue.
select customer_type, sum(total) as total_revenue
from amazon group by customer_type order by total_revenue desc limit 1;

/* This query identifies the customer type that contributes the highest revenue in the dataset. It calculates the total revenue for each customer
type using the SUM(total) function, groups the results by customer type, and then orders them in descending order based on the total revenue. */
```

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

customer_type	total_revenue
Member	164223.44400000002

```sql
-- 17. Determine the city with the highest VAT percentage.
select city, sum(vat) as total_vat, sum(total) as total_sales, (sum(vat) / sum(total)) * 100 as vat_percentage
from amazon group by city order by vat_percentage desc limit 1;

/*
This query determines the city with the highest Value Added Tax (VAT) percentage in the dataset. It calculates the total VAT and total sales for
each city using the SUM(vat) and SUM(total) functions, respectively. Then, it calculates the VAT percentage for each city by dividing the total VAT
by the total sales and multiplying by 100. The results are then ordered in descending order based on the VAT percentage*/
```

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

city	total_vat	total_sales	vat_percentage
Mandalay	5057.032000000003	106197.67199999996	4.761904761904766

Execute the selected portion of the script or everything, if there is no selection

```sql
-- 18. Identify the customer type with the highest VAT payments.
select customer_type, sum(vat) as total_vat_payments
from amazon group by customer_type order by total_vat_payments desc limit 1;

/*This query identifies the customer type that has the highest VAT payments in the dataset. It calculates the total VAT payments for each customer type
using the SUM(vat) function, groups the results by customer type, and then orders them in descending order based on the total VAT payments.*/
```

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

customer_type	total_vat_payments
Member	7820.164000000002

```sql
-- 19. What is the count of distinct customer types in the dataset?
select count(distinct customer_type) as distinct_customer_types from amazon;

/*
This query calculates the count of distinct customer types in the dataset. It uses the COUNT(DISTINCT customer_type) function to count the number of
unique customer types in the "amazon" dataset and assigns the result to a column named distinct_customer_types.*/
```

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

distinct_customer_types
2

```sql
-- 20. What is the count of distinct payment methods in the dataset?
select count(distinct payment) as distinct_payment_methods from amazon;

/*This query calculates the count of distinct payment methods in the dataset. It uses the COUNT(DISTINCT payment) function to count the number of
unique payment methods in the "amazon" dataset and assigns the result to a column named distinct_payment_methods.*/
```

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

distinct_payment_methods
3

```sql
-- 21. Which customer type occurs most frequently?
select customer_type, count(*) as frequency
from amazon group by customer_type order by frequency desc limit 1;

/*
This query identifies the customer type that occurs most frequently in the dataset. It counts the occurrences of each customer type using the
COUNT(*) function, groups the results by customer type, and then orders them in descending order based on the frequency.*/
```

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

customer_type	frequency
Member	501

AmazonSQL* | Limit to 1000 rows

```sql
-- 22. Identify the customer type with the highest purchase frequency.
select customer_type, count(*) as purchase_frequency
from amazon group by customer_type order by purchase_frequency desc limit 1;

/*This query identifies the customer type with the highest purchase frequency in the dataset. It counts the number of purchases made by each customer type using
the COUNT(*) function, groups the results by customer type, and then orders them in descending order based on the purchase frequency.*/
```

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

customer_type	purchase_frequency
Member	501

```sql
-- 23. Determine the predominant gender among customers.
select gender, count(*) as customer_count
from amazon group by gender order by customer_count desc limit 1;

/*
This query determines the predominant gender among customers in the dataset. It counts the occurrences of each gender
using the COUNT(*) function, groups the results by gender, and then orders them in descending order based on the customer count.*/
```

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

gender	customer_count
Female	501

```sql
-- 24. Examine the distribution of genders within each branch.
select branch, gender, count(*) as gender_count
from amazon group by branch, gender order by branch, gender;

/*
This query examines the distribution of genders within each branch in the dataset. It counts the occurrences of each gender within each branch
using the COUNT(*) function, groups the results by branch and gender, and then orders them first by branch and then by gender.*/
```

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

branch	gender	gender_count
A	Female	161
A	Male	179
B	Female	162
B	Male	170
C	Female	178
C	Male	150

```sql
-- 25. Identify the time of day when customers provide the most ratings.
select hour(time) as hour_of_day,
count(*) as rating_count from amazon where rating is not null group by hour(time) order by rating_count desc limit 1;

/*This query identifies the time of day when customers provide the most ratings in the dataset. It extracts the hour of the day from the time column
using the HOUR(time) function, counts the occurrences of ratings (where the rating is not null), groups the results by the hour of the day, and then orders
them in descending order based on the rating count.*/
```

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

hour_of_day	rating_count
19	113

```sql
-- 26. Determine the time of day with the highest customer ratings for each branch.
select branch, hour(time) as hour_of_day,
avg(rating) as average_rating from amazon where rating is not null group by branch, hour(time) order by branch, average_rating desc;

/* This query determines the time of day with the highest customer ratings for each branch in the dataset. It extracts the hour of the day from the
time column using the HOUR(time) function, calculates the average rating (excluding null ratings), groups the results by branch and hour of the day,
and then orders them first by branch and then by average rating in descending order.*/
```

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

branch	hour_of_day	average_rating
A	18	7.318181818181818
A	13	7.3
A	15	7.245945945945945
A	12	7.163636363636363
A	10	7.1578947368421035
A	14	7
A	19	6.851851851851852
A	17	6.84074074074074
A	11	6.840000000000001
A	16	6.718749999999998
A	20	6.627272727272725
B	12	7.396000000000002
B	20	7.01923076923077
B	18	6.974285714285714
B	10	6.915384615384615
B	11	6.872727272727272
B	14	6.783333333333333
B	17	6.74
B	13	6.71578947368421
B	16	6.664705882352941

Result 33

```sql
-- 27. Identify the day of the week with the highest average ratings.
select dayname(date) as day_of_week, avg(rating) as average_rating
from amazon where rating is not null group by dayofweek(date) order by average_rating desc limit 1;

/* This query identifies the day of the week with the highest average ratings in the dataset. It extracts the day of the week from the date column using the
DAYNAME(date) function, calculates the average rating (excluding null ratings), groups the results by the day of the week, and then orders them in descending
order based on the average rating.*/
```

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

day_of_week	average_rating
Monday	7.153599999999999

```sql
-- 28. Determine the day of the week with the highest average ratings for each branch.
select branch, dayname(date) as day_of_week, avg(rating) as average_rating
from amazon where rating is not null group by branch, dayofweek(date) order by branch, average_rating desc;

/* This query determines the day of the week with the highest average ratings for each branch in the dataset. It extracts the branch, day of the week from t
date column using the DAYNAME(date) function, calculates the average rating (excluding null ratings), groups the results by branch and day of the week, and
then orders them first by branch and then by average rating in descending order.*/
```

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

branch	day_of_week	average_rating
A	Friday	7.3119999999999985
A	Monday	7.097916666666666
A	Sunday	7.078846153846157
A	Tuesday	7.0588235294117645
A	Thursday	6.958695652173914
A	Wednesday	6.916279069767441
A	Saturday	6.745999999999998
B	Monday	7.335897435897434
B	Tuesday	7.001886792452827
B	Sunday	6.888571428571429
B	Thursday	6.752272727272726
B	Saturday	6.7366666666666655
B	Friday	6.694117647058826
B	Wednesday	6.451999999999999
C	Friday	7.278947368421051
C	Saturday	7.229629629629631
C	Wednesday	7.064000000000004
C	Monday	7.036842105263159
C	Sunday	7.02826086956522
C	Tuesday	6.95185185185185

Result 35