Key Insights - SQL File

```
1------Select the first 10 rows of the Transactions Table.
select * from transactions LIMIT 10;

2------Select the last 10 rows of the Transactions Table.
select * from transactions ORDER by invoice_id desc LIMIT 10;

3-------List all unique Payment_Method values and their counts.
select DISTINCT(payment_method) from transactions

4-------List count of unique Payment_Method values
select DISTINCT(payment_method),count(payment_method) from transactions
GROUP by payment_method;

5-----Find the total number of transactions in the dataset.
SELECT count(distinct(transaction_id)) from transactions;

6---------Show all transactions where Transaction_Status = 'Refunded' and Total_Sales > 500
select * from transactions where transaction_status='Refunded' and total_sales>500;


7-----Return `Customer_ID`, `Total_Sales` for transactions sorted by highest `Total_Sales`

SELECT customer_id,total_sales from transactions order by total_sales desc limit 20

8-----Select customers with `Loyalty_Points > 3000` and show `Customer_ID`, `Customer_Name`

select customer_id,customer_name,loyalty_points from customer where loyalty_points>3000;

9-----Count number of customers per `Customer_Segment`.

SELECT customer_segment,count(customer_id) from customer group by customer_segment

10-------Get average `Total_Sales` for all transactions status.

SELECT transaction_status,avg(total_sales) from transactions group by transaction_status;

11------Find the maximum `Quantity` sold in a single transaction and show that transaction

SELECT transaction_id,max(quantity) from transactions group by transaction_id order by quan


12------Return the number of distinct `Customer_ID`s who made purchases (i.e., active custo

SELECT count(DISTINCT(customer_id)) from customer


13-----Join Transactions Table with Customers Table to show Customer_Name, Tranasaction_Dat
SELECT customer_name,tranasaction_date,total_sales from customer c INNER join transactions
```

```
14-----For each Customer_Segment, compute total revenue and average order value.

select customer_segment,
■sum(total_sales*quantity),
■round(avg(total_sales*quantity),1)
■from customer c
■inner join transactions t on c.customer_id = t.customer_id
■group by customer_segment;

--15-------Find top 10 products by total revenue (use Product Table + Transactions Table).

select p.product_name,sum(total_sales*quantity) as total_revenue from product p
inner join transactions t on p.product_id = t.transaction_id
GROUP by p.product_name
order by total_revenue desc
limit 10;


--16------Identify customers who have more than 5 transactions. Show `Customer_ID` and `txn

select customer_id, count(*)
from transactions
group by customer_id
having count(*)>5;

--17-----For each `Payment_Method`, compute success rate: (#Success / total). Show `Payment

SELECT payment_method,
sum(case when transaction_status='Success' then 1 else 0 END)*100/count(*) || '%'
FROM transactions
group by payment_method;


--18------List invoices that are overdue assuming Payment_Due = 'NO' means unpaid. Show inv
SELECT i.invoice_id,c.customer_id,i.invoice_date,i.payment_due from invoice i
inner join transactions t on i.invoice_id = t.invoice_id
inner join customer c on t.customer_id = c.customer_id
WHERE i.payment_due='NO'

--19--Create a column is_high_value in a query: 1 if Total_Sales >= 1000 else 0. Show count
select customer_id,
sum(case when total_sales>=1000 then 1 else 0 end) is_high_value
from transactions
group by customer_id;

--20--10. Find products where average discount per transaction > 20. Show `Product_ID`, `av

   -- Hint: `AVG(Discount)` grouped by `Product_ID`, filter with `HAVING`.
```

```
SELECT t.product_id,p.product_name,round(avg(discount),2)
from transactions t
inner join product p
on t.product_id = p.product_id
group by t.product_id,p.product_name
having avg(discount)>=20
```