

E-Commerce Data Analysis Project

SQL Questions & Answers

BASIC PROBLEMS

1. List all unique cities where customers are located.

SQL:

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

2. Count the number of orders placed in 2017.

SQL:

```
SELECT COUNT(*) FROM orders WHERE YEAR(order_date)=2017;
```

3. Find the total sales per product category.

SQL:

```
SELECT p.category, SUM(oi.price) FROM order_items oi JOIN products p ON  
oi.product_id=p.product_id GROUP BY p.category;
```

4. Percentage of orders paid in installments.

SQL:

```
SELECT COUNT(CASE WHEN payment_installments>1 THEN 1 END)*100/COUNT(*) FROM  
payments;
```

5. Count customers from each state.

SQL:

```
SELECT customer_state, COUNT(*) FROM customers GROUP BY customer_state;
```

INTERMEDIATE PROBLEMS

6. Orders per month in 2018.

SQL:

```
SELECT MONTH(order_date), COUNT(*) FROM orders WHERE YEAR(order_date)=2018  
GROUP BY MONTH(order_date);
```

7. Average products per order by city.

SQL:

```
SELECT c.customer_city, AVG(cnt) FROM (SELECT o.order_id, c.customer_city, COUNT(*) cnt  
FROM orders o JOIN customers c ON o.customer_id=c.customer_id JOIN order_items oi ON  
o.order_id=oi.order_id GROUP BY o.order_id, c.customer_city) t GROUP BY customer_city;
```

8. Revenue percentage by category.

SQL:

```
SELECT p.category, SUM(oi.price)*100/(SELECT SUM(price) FROM order_items) FROM  
order_items oi JOIN products p ON oi.product_id=p.product_id GROUP BY p.category;
```

9. Correlation between price and purchase count (data prep).

SQL:

```
SELECT product_id, COUNT(*), AVG(price) FROM order_items GROUP BY product_id;
```

10. Seller revenue and ranking.

SQL:

```
SELECT seller_id, SUM(price), RANK() OVER(ORDER BY SUM(price) DESC) FROM order_items  
GROUP BY seller_id;
```

ADVANCED PROBLEMS

11. Moving average of order value per customer.

SQL:

```
SELECT customer_id, order_date, AVG(order_value) OVER(PARTITION BY customer_id ORDER BY order_date ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) FROM (SELECT o.customer_id, o.order_date, SUM(oi.price) order_value FROM orders o JOIN order_items oi ON o.order_id=oi.order_id GROUP BY o.order_id) t;
```

12. Cumulative sales per month per year.

SQL:

```
SELECT year, month, SUM(monthly_sales) OVER(PARTITION BY year ORDER BY month) FROM (SELECT YEAR(o.order_date) year, MONTH(o.order_date) month, SUM(oi.price) monthly_sales FROM orders o JOIN order_items oi ON o.order_id=oi.order_id GROUP BY year, month) t;
```

13. Year-over-Year growth.

SQL:

```
SELECT year, total_sales, ((total_sales-LAG(total_sales) OVER(ORDER BY year))/LAG(total_sales) OVER(ORDER BY year))*100 FROM (SELECT YEAR(o.order_date) year, SUM(oi.price) total_sales FROM orders o JOIN order_items oi ON o.order_id=oi.order_id GROUP BY year) t;
```

14. Customer retention within 6 months.

SQL:

```
SELECT COUNT(DISTINCT o2.customer_id)*100/COUNT(DISTINCT o1.customer_id) FROM orders o1 LEFT JOIN orders o2 ON o1.customer_id=o2.customer_id AND o2.order_date BETWEEN o1.order_date AND DATE_ADD(o1.order_date, INTERVAL 6 MONTH);
```

15. Top 3 customers by spending each year.

SQL:

```
SELECT * FROM (SELECT YEAR(o.order_date) year, o.customer_id, SUM(oi.price), RANK() OVER(PARTITION BY YEAR(o.order_date) ORDER BY SUM(oi.price) DESC) r FROM orders o JOIN order_items oi ON o.order_id=oi.order_id GROUP BY year, o.customer_id) t WHERE r<=3;
```