PostgreSQL Practice Questions - SQL Fundamentals

Basic Level Questions (1-20)

1. Write a query to select all columns from a table named `employees`.
2. Select only the `first_name` and `last_name` columns from the `employees` table.
3. Create a query that selects the `product_name` column and gives it an alias `Product`.
4. Write a query to find all employees where the `salary` is exactly 50000.
5. Select all products where the `price` is greater than 100.
6. Find all customers whose `age` is less than or equal to 30.
7. Write a query to select employees whose `department` is not equal to 'HR'.
8. Select all orders where the `order_date` is between '2023-01-01' and '2023-12-31'.
9. Find all products where the `category` is either 'Electronics', 'Books', or 'Clothing'.
10. Write a query to select customers whose `city` is not in ('New York', 'Los Angeles', 'Chicago').
11. Find all employees whose `first_name` starts with 'J'.
12. Select products where the `product_name` contains the word 'Pro'.
13. Write a query to find employees whose `email` ends with '@company.com'.
14. Select all records from the `sales` table and sort them by `sale_amount` in ascending order.
15. Find all customers sorted by `last_name` in descending order.
16. Write a query to select the first 10 records from the `products` table.
17. Select records 11-20 from the `employees` table (using OFFSET).
18. Find all orders where the `status` is 'Pending' and sort by `order_date`.
19. Write a single-line comment explaining what your query does, then select all from `customers`.
20. Create a multi-line comment block and write a query to select `product_id` and `price` from `products`.

Intermediate Level Questions (21-40)

21. Select employees whose `salary` is between 40000 and 80000 and `department` is 'IT'.
22. Find products where `price` is greater than 50 OR `category` is 'Electronics', sorted by `price` DESC.
23. Write a query to select customers where `age` is NOT between 25 and 65.
24. Find employees whose `first_name` starts with 'A' AND `last_name` ends with 'son'.
25. Select orders where `order_total` is greater than 1000 OR `customer_id` is in (1, 5, 10).

26. Write a query using ILIKE to find products where `description` contains 'wireless' (case-insensitive).
27. Find all employees except those in departments 'HR', 'Finance', and 'Legal'.
28. Select the top 5 highest-paid employees, showing only `name` and `salary`.
29. Write a query to find customers whose `phone_number` follows the pattern '555--'.
30. Select products where `stock_quantity` is less than 10 AND `price` is greater than 20.
31. Find employees hired in 2023, sorted by `hire_date` DESC, limit to first 15 records.
32. Write a query to select orders where `shipping_address` does NOT contain 'PO Box'.
33. Find customers where `email` is NOT NULL and `city` is either 'Boston' or 'Seattle'.
34. Select products with `rating` between 4.0 and 5.0, excluding category 'Books'.
35. Write a query to find employees whose `manager_id` is NULL (top-level managers).
36. Select orders from the last quarter of 2023, sorted by `order_date` and `customer_id`.
37. Find products where `brand` starts with 'Apple' OR `brand` starts with 'Samsung'.
38. Write a query to select customers, skip the first 25 records, and show the next 10.
39. Find employees where `salary` is greater than 60000 AND (`department` is 'Sales' OR `department` is 'Marketing').
40. Select all orders where `discount_percent` is between 10 and 30, showing `order_id`, `total` as `Order_Total`.

Advanced Level Questions (41-60)

41. Write a query to find products where `price` is above average, but you cannot use subqueries. Instead, assume average price is 75 and use logical operators.
42. Find customers who live in cities starting with 'San' and have made orders worth more than 500 (assume `order_total` column exists in customers table).
43. Select employees whose `full_name` (assuming it's a single column) contains both 'John' and 'Smith' using pattern matching.
44. Write a query to find the most expensive product in each category, but limit results to top 3 categories by average price.
45. Find all orders placed on weekends in January 2024 (assume `order_date` contains day information).
46. Select products where `title` matches the pattern '% Pro %' but NOT '% Pro Max %'.
47. Write a query to find employees with salaries in the top 10% (assume 90th percentile is 90000).
48. Find customers whose `registration_date` is NOT in the first or last week of any month.
49. Select orders where `item_count` is odd numbers between 5 and 15.

50. Write a query to find products with names containing numbers, sorted by the numeric value within the name.
51. Find employees where `performance_rating` is 'A' OR 'B' AND `years_experience` is more than 5.
52. Select customers from specific zip code ranges: 10001-19999, 20001-29999, or 90001-99999.
53. Write a query to find orders where `special_instructions` is either NULL or contains specific keywords.
54. Find products where `launch_date` is within the last 2 years but not in the current year.
55. Select employees whose `employee_id` is divisible by 3 and `department_code` starts with 'D'.
56. Write a query to find overlapping date ranges in a `projects` table using date comparisons.
57. Find customers where the domain of their `email` is NOT from common providers (gmail, yahoo, hotmail).
58. Select products where `dimensions` (stored as text) contains measurements between 10-50 for any dimension.
59. Write a query to find employees with consecutive `employee_id` numbers who work in the same department.
60. Find orders where multiple conditions create complex logical groupings: ((A AND B) OR (C AND D)) AND NOT E.

Expert/Interview Level Questions (61-80)

61. **Case Study**: You have an e-commerce database. Find products that are popular (sold more than 100 times) but currently low in stock (less than 20 units), excluding discontinued items. Sort by urgency (stock level ascending, then sales volume descending). Show product name as "Product", current stock as "Urgent_Restock_Needed".
62. **Performance Scenario**: Write a query to find customers who haven't placed orders in the last 6 months. The customer table has 1M+ records. Structure your WHERE clause for optimal performance using date comparisons and NOT EXISTS logic simulation.
63. **Data Quality Check**: Find employee records where data integrity might be compromised: salary is 0 or NULL, hire_date is in the future, or email format is invalid (doesn't contain @ symbol).
64. **Business Logic**: In a subscription service, find users whose subscription expires within 30 days, but exclude those who've already been notified (notification_sent = true) or are on annual plans with auto-renewal enabled.
65. **Complex Filtering**: From a sales database, find transactions where the discount percentage seems suspicious: either too high (>50%) for non-clearance items, or applied to already discounted products (original_price != current_price).
66. **Inventory Management**: Find products that need reordering based on complex criteria: (current_stock < reorder_level) OR (projected_demand > current_stock * 1.5) OR (supplier_lead_time

> 30 days AND current_stock < 50).

67. **Customer Segmentation**: Identify VIP customers using multiple criteria: total_orders > 20 OR total_spent > 10000 OR (customer_since < '2020-01-01' AND average_order_value > 500).
68. **Fraud Detection**: Find potentially fraudulent orders: large order values (>5000) from new customers (registered < 30 days ago) OR multiple orders from same IP but different customer accounts OR orders with expedited shipping to high-risk zip codes.
69. **Seasonal Analysis**: Find products that show seasonality patterns: high sales in Q4 (Oct-Dec) but low sales in Q1-Q3, excluding holiday-specific categories and limited-time offers.
70. **Supply Chain**: Identify suppliers with delivery issues: average_delivery_time > promised_delivery_time OR on_time_percentage < 85% OR (recent_delays > 3 AND contract_start_date > '2023-01-01').
71. **User Behavior**: Find users showing churn indicators: last_login > 60 days ago AND (subscription_type = 'free' OR payment_failed = true) AND email_engagement_score < 20.
72. **Price Optimization**: Find products where pricing might need adjustment: competitor_price differs by more than 15% from our_price OR (sales_velocity < 0.5 AND price_last_updated < 6 months ago).
73. **A/B Testing**: Identify users for a test group: registered between specific dates, from target demographics (age 25-45), with medium engagement levels (not too high, not too low), excluding previous test participants.
74. **Compliance Check**: Find records that might violate data retention policies: personal_data_collected = true AND (last_activity_date < '2021-01-01' OR consent_withdrawn = true OR user_status = 'inactive').
75. **Marketing Campaign**: Select customers for a re-engagement campaign: purchased before but not in last 6 months, email_subscribed = true, previous_campaign_response_rate > 10%, and not in current exclusion list.
76. **Quality Assurance**: Find data inconsistencies: orders where order_total != (item_price * quantity - discount + tax + shipping) OR customer_age < 18 but account_type = 'adult' OR phone_number contains invalid characters.
77. **Resource Planning**: Identify peak usage periods: concurrent_users > 1000 AND (server_response_time > 2 seconds OR error_rate > 5%) AND time_of_day BETWEEN '09:00' AND '17:00'.
78. **Risk Assessment**: Find high-risk accounts: multiple_failed_logins > 5 OR login_from_new_location = true OR (account_value > 50000 AND security_questions_answered < 3).
79. **Operational Efficiency**: Identify process bottlenecks: processing_time > average_processing_time * 1.5 OR requires_manual_review = true OR (priority = 'high' AND status = 'pending' AND created_date < current_date - 1).
80. **Data Migration Validation**: Find records that need special handling during migration: contains_special_characters = true OR data_format != 'standard' OR (record_size > 1MB AND compression_supported = false) OR foreign_key_references > 10.

Answer Key Notes:

- Questions 1-20 focus on single concepts
- Questions 21-40 combine 2-3 concepts
- Questions 41-60 use complex combinations with multiple operators
- Questions 61-80 are real-world scenarios requiring comprehensive understanding
- Each question can be solved using only the fundamental concepts listed
- No advanced SQL features (joins, subqueries, functions) are required
- Practice writing actual SQL for each question
- Pay attention to operator precedence in complex WHERE clauses
- Remember to use proper semicolon terminators and comments as needed