IT Skill Test GIC Myanmar

Duration: 30 Minutes Total Questions: 8

---

1. You need to create a view that combines data from the EMPLOYEES and DEPARTMENTS tables, showing employee names and their department names. Which SQL statement would you use?

   A.CREATE VIEW emp_dept_view AS SELECT e.first_name, e.last_name, d.department_name FROM employees e, departments d;

   B.CREATE VIEW emp_dept_view AS SELECT e.first_name, e.last_name, d.department_name FROM employees e JOIN departments d ON e.department_id = d.department_id;

   C.CREATE VIEW emp_dept_view AS SELECT first_name, last_name, department_name FROM employees, departments;

   D.CREATE VIEW emp_dept_view (first_name, last_name, department_name) AS SELECT * FROM employees, departments;

2. You have a view named SALES_SUMMARY that needs to be updated to include a new column for total sales. What is the correct SQL statement to modify this existing view?

   A.ALTER VIEW sales_summary ADD COLUMN total_sales NUMBER;

   B.MODIFY VIEW sales_summary ADD (total_sales NUMBER);

   C.CREATE OR REPLACE VIEW sales_summary AS SELECT existing_columns, SUM(amount) AS total_sales FROM original_table GROUP BY existing_columns;

   D.UPDATE VIEW sales_summary SET total_sales = SUM(amount);

3. You need to create a view that displays the top 10 customers by order value. Which of the following SQL statements would correctly create this view?

   A. CREATE VIEW top_10_customers AS SELECT customer_id, SUM(order_total) AS total_value FROM orders GROUP BY customer_id ORDER BY total_value DESC LIMIT 10;

   B. CREATE VIEW top_10_customers AS SELECT customer_id, SUM(order_total) AS total_value FROM orders GROUP BY customer_id ORDER BY total_value DESC FETCH FIRST 10 ROWS ONLY;

   C. CREATE VIEW top_10_customers AS SELECT customer_id, SUM(order_total) AS total_value FROM orders GROUP BY customer_id ORDER BY total_value DESC WHERE ROWNUM <= 10;

   D. CREATE VIEW top_10_customers AS SELECT TOP 10 customer_id, SUM(order_total) AS total_value FROM orders GROUP BY customer_id ORDER BY total_value DESC;

4. You need to create a view that combines data from the EMPLOYEES and DEPARTMENTS tables, but you want to ensure that users cannot see salary information for departments they don't have permission to access. Which approach should you use?

   A. CREATE VIEW emp_dept_secure AS SELECT e.employee_id, e.first_name, e.last_name, d.department_name, e.salary FROM employees e JOIN departments d ON e.department_id = d.department_id WHERE USER = d.manager_id;

   B. CREATE VIEW emp_dept_secure AS SELECT e.employee_id, e.first_name, e.last_name, d.department_name, CASE WHEN USER = d.manager_id THEN e.salary ELSE NULL END AS salary FROM employees e JOIN departments d ON e.department_id = d.department_id;

   C. CREATE VIEW emp_dept_secure AS SELECT e.employee_id, e.first_name, e.last_name, d.department_name, e.salary FROM employees e JOIN departments d ON e.department_id = d.department_id WITH CHECK OPTION;

   D. CREATE VIEW emp_dept_secure AS SELECT e.employee_id, e.first_name, e.last_name, d.department_name, DECODE(USER, d.manager_id, e.salary, NULL) AS salary FROM employees e JOIN departments d ON e.department_id

= d.department_id;

5. You need to create a view that will be used for reporting purposes, combining data from the ORDERS and ORDER_ITEMS tables. The view should include the order ID, customer ID, order date, and total order amount. Which of the following SQL statements would correctly create this view?

   A.CREATE VIEW order_summary AS SELECT o.order_id, o.customer_id, o.order_date, SUM(oi.quantity * oi.unit_price) AS total_amount FROM orders o, order_items oi GROUP BY o.order_id, o.customer_id, o.order_date;

   B.CREATE VIEW order_summary AS SELECT o.order_id, o.customer_id, o.order_date, SUM(oi.quantity * oi.unit_price) AS total_amount FROM orders o JOIN order_items oi ON o.order_id = oi.order_id GROUP BY o.order_id, o.customer_id, o.order_date;

   C.CREATE VIEW order_summary AS SELECT order_id, customer_id, order_date, (SELECT SUM(quantity * unit_price) FROM order_items WHERE order_id = orders.order_id) AS total_amount FROM orders;

   D.CREATE VIEW order_summary AS SELECT o.order_id, o.customer_id, o.order_date, oi.total_amount FROM orders o, (SELECT order_id, SUM(quantity * unit_price) AS total_amount FROM order_items GROUP BY order_id) oi WHERE o.order_id = oi.order_id;

6. You are tasked with creating a view that displays the total sales for each product category in the last quarter. The view should include the category name, total sales amount, and the percentage of total sales across all categories. Which SQL query would correctly create this view?

   A.CREATE VIEW category_sales AS
   SELECT c.category_name, SUM(s.amount) AS total_sales,
         SUM(s.amount) / SUM(SUM(s.amount)) OVER () * 100 AS
   sales_percentage
   FROM sales s
   JOIN products p ON s.product_id = p.product_id
   JOIN categories c ON p.category_id = c.category_id

```
 WHERE s.sale_date >= ADD_MONTHS(TRUNC(SYSDATE, 'Q'), -3)
 GROUP BY c.category_name;
```

B.CREATE VIEW category_sales AS

```
SELECT c.category_name, SUM(s.amount) AS total_sales,
       ROUND(SUM(s.amount) / SUM(SUM(s.amount)) OVER () * 100, 2) AS
sales_percentage
FROM sales s
JOIN products p ON s.product_id = p.product_id
JOIN categories c ON p.category_id = c.category_id
WHERE s.sale_date >= ADD_MONTHS(TRUNC(SYSDATE, 'Q'), -3)
GROUP BY c.category_name;
```

C.CREATE VIEW category_sales AS

```
SELECT c.category_name, SUM(s.amount) AS total_sales,
       ROUND(SUM(s.amount) / (SELECT SUM(amount) FROM sales) * 100, 2) AS
sales_percentage
FROM sales s
JOIN products p ON s.product_id = p.product_id
JOIN categories c ON p.category_id = c.category_id
WHERE s.sale_date >= ADD_MONTHS(TRUNC(SYSDATE, 'Q'), -3)
GROUP BY c.category_name;
```

D.CREATE VIEW category_sales AS

```
SELECT c.category_name, SUM(s.amount) AS total_sales,
       ROUND(SUM(s.amount) / SUM(SUM(s.amount)) OVER () * 100, 2) AS
sales_percentage
FROM sales s
JOIN products p ON s.product_id = p.product_id
JOIN categories c ON p.category_id = c.category_id
GROUP BY c.category_name
HAVING s.sale_date >= ADD_MONTHS(TRUNC(SYSDATE, 'Q'), -3);
```

7. Which SQL statement would you use to create a read-only view based on the EMPLOYEES table that only shows active employees (status = 'ACTIVE') and includes their full name as a concatenated field?

A.
```
CREATE VIEW active_employees AS
SELECT employee_id, first_name || ' ' || last_name AS full_name, email,
hire_date
FROM employees
WHERE status = 'ACTIVE'
WITH READ ONLY;
```

B.
```
CREATE OR REPLACE VIEW active_employees AS
SELECT employee_id, CONCAT(first_name, ' ', last_name) AS full_name,
email, hire_date
FROM employees
WHERE status = 'ACTIVE'
WITH CHECK OPTION;
```

C.
```
CREATE VIEW active_employees (employee_id, full_name, email, hire_date)
AS
SELECT employee_id, first_name || ' ' || last_name, email, hire_date
FROM employees
WHERE status = 'ACTIVE'
WITH READ ONLY;
```

D.
```
CREATE OR REPLACE VIEW active_employees AS
SELECT employee_id, first_name || ' ' || last_name AS full_name, email,
hire_date
FROM employees
WHERE status = 'ACTIVE';
```

8. You need to create a view that shows the top 5 customers by total purchase amount for each product category. The view should include the category name, customer name, and total purchase amount. However, you've noticed that the query is running slowly due to the large volume of data. Which of the following approaches would be most effective in improving the performance of this view?

A. CREATE MATERIALIZED VIEW top_customers AS

```
SELECT * FROM (
  SELECT c.category_name, cu.customer_name, SUM(o.total_amount) AS
total_purchases,
         ROW_NUMBER() OVER (PARTITION BY c.category_name ORDER BY
SUM(o.total_amount) DESC) AS rn
  FROM orders o
  JOIN products p ON o.product_id = p.product_id
  JOIN categories c ON p.category_id = c.category_id
  JOIN customers cu ON o.customer_id = cu.customer_id
  GROUP BY c.category_name, cu.customer_name
)
WHERE rn <= 5;
```

B. CREATE VIEW top_customers AS

```
SELECT * FROM (
  SELECT c.category_name, cu.customer_name, SUM(o.total_amount) AS
total_purchases,
         RANK() OVER (PARTITION BY c.category_name ORDER BY SUM(o.total_amount)
DESC) AS rnk
  FROM orders o
  JOIN products p ON o.product_id = p.product_id
  JOIN categories c ON p.category_id = c.category_id
  JOIN customers cu ON o.customer_id = cu.customer_id
  GROUP BY c.category_name, cu.customer_name
)
WHERE rnk <= 5;
```

C. CREATE VIEW top_customers AS

```
SELECT c.category_name, cu.customer_name, SUM(o.total_amount) AS
total_purchases
FROM orders o
JOIN products p ON o.product_id = p.product_id
JOIN categories c ON p.category_id = c.category_id
JOIN customers cu ON o.customer_id = cu.customer_id
GROUP BY c.category_name, cu.customer_name
HAVING SUM(o.total_amount) >= (
    SELECT MIN(total_purchases)
    FROM (
      SELECT c2.category_name, SUM(o2.total_amount) AS total_purchases
      FROM orders o2
      JOIN products p2 ON o2.product_id = p2.product_id
      JOIN categories c2 ON p2.category_id = c2.category_id
      GROUP BY c2.category_name
      ORDER BY SUM(o2.total_amount) DESC
      FETCH FIRST 5 ROWS ONLY
    )
    WHERE category_name = c.category_name
  );
```

D.CREATE OR REPLACE FORCE VIEW top_customers AS

```
  SELECT * FROM (
    SELECT c.category_name, cu.customer_name, SUM(o.total_amount) AS
  total_purchases,
           ROW_NUMBER() OVER (PARTITION BY c.category_name ORDER BY
  SUM(o.total_amount) DESC) AS rn
    FROM orders o
    JOIN products p ON o.product_id = p.product_id
    JOIN categories c ON p.category_id = c.category_id
    JOIN customers cu ON o.customer_id = cu.customer_id
    GROUP BY c.category_name, cu.customer_name
  )
  WHERE rn <= 5;
```