Which statements are true regarding the USING and ON clauses in table joins?

Select one or more:

- a. Both USING and ON clause can be used for equijoins and nonequijoins
- b. Maximum of one pair of columns can be joined between two tables using the ON clause
- c. The WHERE clause can be used to apply additional conditions in SELECT statement containing the ON or the USING clause. ✔
- d. The ON clause can be used to join tables on columns that have different names but compatible data types. ✔

Your answer is correct.

The correct answers are: The ON clause can be used to join tables on columns that have different names but compatible data types., The WHERE clause can be used to apply additional conditions in SELECT statement containing the ON or the USING clause.

To display the names of employees who earns more than the average salary of all employees.

```
SELECT last_name, first_name
FROMemployee
WHEREsalary > AVG(salary);
```

Which change should you make to achieve the desired results?

Select one:

- a. Move the function to the SELECT clause and add a GROUP BY clause.
- b. Change the function in the WHERE clause.
- c. Move the function to the SELECT clause and add a GROUP BY clause and a HAVING clause.
- d. Use a subquery in the WHERE clause to compare the average salary value. ✔

The correct answer is: Use a subquery in the WHERE clause to compare the average salary value.

Consider the below tables:

Promotions Table

Column Name	Datatype	Constraint
Promo_id	Number	PK
Promo_name	Varchar	
Promo_begin_date	Date	
Promo_end_date	Date	

Sales Table

Column Name	Datatype	Constraint
Promo_id	Number	FK
Cust_id	Number	FK
Time_id	Date	

Customer Table

Column Name	Datatype	Constraint
cust_id	Number	PK
cust_name	Varchar	

The Below query will generate a report showing the promo name along with the
customer name for all products that were sold during their promo campaign and before 30th October 2007.

```
SELECT promo_name,cust_name FROM promotions p JOIN sales s

ON(time_id BETWEEN promo_begin_date AND promo_end_date)

JOIN customer c ON (s.cust_id = c.cust_id) AND time_id < '30-oct-2007';
```

Which statement is true regarding the above query?

Select one:

- a. It produces an error because equijoin and nonequijoin conditions cannot be used in the same
- b. It executes successfully and gives the required result.
- c. It produces an error because the join order of the tables is incorrect.
- d. It executes successfully but does not give the required result. ✔

Your answer is correct.

The correct answer is: It executes successfully but does not give the required result.

The COMMISSION column shows the monthly commission earned by the employee.

Emp_Id	Dept_Id	Commission
1	10	500
2	20	1000
3	10	
4	10	600
5	30	800
6	30	200
7	10	
8	20	300

Which tasks would require sub queries or joins in order to be performed in a single step?

Select one or more:

- a. Listing the departments whose average commission is more that 600
- b. Finding the number of employees who earn a commission that is higher than the average commission of the company ✔
- c. Listing the employees who do not earn commission and who are working for department 20 in descending order of the employee ID
- d. Finding the total commission earned by the employees in department 10
- e. Listing the employees whose annual commission is more than 6000
- f. Listing the employees who earn the same amount of commission as employee 3 ✔

Your answer is correct.

The correct answers are: Listing the employees who earn the same amount of commission as employee 3, Finding the number of employees who earn a commission that is higher than the average commission of the company

Which statement would display the highest credit limit available in each income level in each city in the Customers table?

Select one:

- a. SELECT cust_city, cust_income_level,MAX(cust_credit_limit)
  FROM customers
  GROUP BY cust_city , , cust_income_level ,MAX(cust_credit_limit);
- b. SELECT cust_city, cust_income_level,MAX(cust_credit_limit) ✔
  FROM customers
  GROUP BY cust_city, cust_income_level;
- c. SELECT cust_city, cust_income_level,MAX(cust_credit_limit)
  FROM customers
  GROUP BY cust_city, cust_income_level,cust_credit_limit;
- d. SELECT cust_city, cust_income_level,MAX(cust_credit_limit)
  FROM customers
  GROUP BY cust_credit_limit , cust_income_level, cust_city ;

The correct answer is: SELECT cust_city, cust_income_level,MAX(cust_credit_limit)
FROM customers
GROUP BY cust_city, cust_income_level;

The following query is written to retrieve all those product IDs from the SALES table that have more than 55000 sold and have been ordered more than 10 times:

SELECT prod_id FROM sales WHERE quantity_sold > 55000 AND COUNT(*)>10

GROUP BY prod_id HAVING COUNT(*)>10;

Which statement is true regarding this SQL statement?

Select one:

- a. It executes successfully but produces no result because COUNT(prod_id) should be used instead of COUNT(*) .
- b. It executes successfully and generates the required result.
- c. It produces an error because COUNT (*) should be only in the HAVING clause and not in the WHERE clause. ✔
- d. It produces an error because COUNT (*) should be specified the SELECT clause also.

The correct answer is: It produces an error because COUNT (*) should be only in the HAVING clause and not in the WHERE clause.

Which statements would execute successfully?

Select one or more:

- a. SELECT student_name,SUM(subject1)
  FROM marks
  WHERE student_name LIKE 'R%';
- b. SELECT SUM (DISTINCT NVL(subject1,0)),MAX(subject1) ✔
  FROM marks
  WHERE subject1 > subject2;
- c. SELECT SUM (subject1+subject2+subject3) ✔
  FROM marks
  WHERE student_name IS NULL
- d. SELECT student_name,subject1
  FROM marks
  WHERE subject1 > AVG(subject1);

The correct answers are: SELECT SUM (DISTINCT NVL(subject1,0)),MAX(subject1)
FROM marks
WHERE subject1 > subject2;, SELECT SUM (subject1+subject2+subject3)
FROM marks
WHERE student_name IS NULL

Which SQL statement produces an error?

Select one:

a.
```
SELECT job_id, SUM(salary)
FROM emp_dept_vu
WHERE department_id IN (10,20)
GROUP BY job_id
HAVING SUM(salary) > 20000;
```

b.
```
SELECT department_id, SUM(salary)
FROM emp_dept_vu
GROUP BY department_id;
```

c. None of the statements produce an error; all are valid. ✔

d.
```
SELECT department_id, job_id, AVG(salary)
FROM emp_dept_vu
GROUP BY department_id, job_id;
```

e.
```
SELECT *
FROM emp_dept_vu;
```

The correct answer is: None of the statements produce an error; all are valid.

```
SELECT cust_city, COUNT(cust_last_name)
FROM customers
WHERE cust_credit_limit > 1000
GROUP BY cust_city
HAVING AVG(cust_credit_limit) BETWEEN 5000 AND 6000;
```

Which statement is true regarding the outcome of the above query?

Select one:

a. It returns an error because WHERE and HAVING clauses cannot be used in the same SELECT statement.

b. It returns an error because the BETWEEN operator cannot be used in the HAVING clause.

c. Date functions

d. It returns an error because WHERE and HAVING clauses cannot be used to apply conditions on the same column.

e. It executes successfully. ✔

Your answer is correct.

The correct answer is: It executes successfully.

To create a report displaying employee last names, department names, and locations. Which query should you use to create an equi-join?

Select one:

a. SELECT e.last_name, d.department_name, d.location_id
FROM employees e, departments d
WHERE manager_id =manager_id;

b. SELECT last_name, department_name, location_id
FROM employees , departments ;

c. SELECT e.last_name, d.department_name, d.location_id ✔
FROM employees e, departments d
WHERE e.department_id =d.department_id;

d. SELECT employees.last_name, departments.department_name,
departments.location_id FROM employees e, departments d
WHERE e.department_id =d.department_id;

Your answer is correct.

The correct answer is: SELECT e.last_name, d.department_name, d.location_id
FROM employees e, departments d
WHERE e.department_id =d.department_id;