|  |  |  |
| --- | --- | --- |
| 6.2 Dates  Sometimes you need to use a date as a search condition. Oracle displays dates in the default format DDMON-YY, with MON being the standard three- letter abbreviation for the month. Because the date field contains letters and hyphens, it's not considered a numeric value when Oracle performs searches.  Therefore, a date value must be enclosed in single quotation marks.  --When filtering for dates, can use the normal operators.  SELECT fname, Iname, DOB, salary FROM patient WHERE DOB>' Il-FEB-1978'  --1f the date format is not default, then to\_date has to be used.  SELECT fname, Iname, DOB, salary FROM patient WHERE DOB>TO DATE ( '02/11/1978' , t m/dd/yyyy') ;  Example 6.6a (Any)  -- Since the values overlap, the following is logically not correct.  SELECT fname, 1 name, salary FROM patient WHERE salary > 30000 OR salary>40000;  --This accomplishes the same thing as above.  SELECT fname, Inane, salary FROM patient WHERE salary 40000) ; | 6.3 LOWER, UPPER  The upper function converts all letters in the specified string to uppercase. If there are characters in the string that are not letters, they are unaffected by this function. The syntax for the upper function is:upper( stringl )stringl is the string to convert to uppercase.  The lower function converts all letters in the specified string to lowercase. If there are characters in the string that are not letters, they are unaffected by this function. The syntax for the lower function is: lower(stringl) stringl is the string to convert to lowercase.  --These examples are to indicate that the data in the table is case sensitive.  SELECT fname, 1 name, city FROM patient WHERE city—' davist ; SELECT fname, 1 name, city FROM patient WHERE city—' Davis' ;  --To bypass case-sensitivity, use the upper or lower function. When using  —upper, make sure that it is being compared with something that is in upper case and the --opposite when checking for lower case.  SELECT fname, Iname,city FROM patient WHERE UPPER (city) -'DAVIS' •  SELECT fname, 1 name, city FROM patient WHERE LOWER (city) ='davis' • | Example 6.4a (The and clause)  At times, you need to search for records based on two or more conditions. In these situations, you can use logical operators to combine search conditions. The logical operators AND and OR are commonly used for this purpose. ( The NOT operator is also a logical operator in Oracle , but it's used to reverse the meaning of search conditions rather than combine them.) Keep in mind that when a query executes, records can be filtered with WHERE clause conditions. If the condition is TRUE when compared with a record, the record is included in the results. When the AND operator is used in the WHERE clause, both conditions combined by the AND operator must be evaluated as  TRUE, or the record isn't included in the results.  Can use the and operator to check for two conditions. Checks for a range in this case.  SELECT fname, Iname, DOB, salary FROM patient WHERE salary>30000 and gal ary <80000;  SELECT fname, Iname, DOB, salary FROM patient WHERE DOB>'11-FEB-1978' and DOB<' 15-MAR-1990' ;  SELECT fname, Iname, DOB, salary, city FROM patient WHERE salary>30000 and LOWER (city) = ' davis' ;  SELECT fname, Iname, DOB, salary, city FROM patient WHERE LOWER (city) —'davis' and LOWER (city) z' reno' ;  —What is wrong?? Notice that the ranges overlap which makes it logically invalid.  SELECT fname, Inane, DOB, salary FROM patient WHERE salary>30000 and salary >80000; |
| Example 6.4b (Betrveen)  The two values defining the range for SQL BETWEEN clause can be dates, numbers or just text. (Inclusive of the end points)  --When looking for a range using the and clause, if it is inclusive of the end points then it -- is easier to use the between clause instead.  SELECT fname, Inane, DOB, salary FROM patient WHERE AND  salary  SELECT fname, Iname, DOB, salary FROM patient WHERE salary BETWEEN 30000 AND 80000; | Example 6.5a (Or, In)  The IN o erator allows ou to s ecif multi le values in a WHERE clause.  --lf the same column is being checked against multiple values, then the IN clause is an easier route.  SELECT fname, Inane, city FROM patient WHERE city—' Davis' OR city=' Sacramento' OR city= ' Chico' ;  —Same as above  SELECT fname, Iname, city FROM patient WHERE city IN  ( 'Davis' , 'Sacramento' , 'Chico') ;  SELECT fname, Jname, salary FROM patient WHERE salary=10000 OR salary=20000 OR salary=30000;  SELECT fname, 1 name, salary FROM patient WHERE salary (10000, 20000, 30000) ;  --Negates the IN clause using the NOT operator which means it displays the record as  —long as the salary is not one of the three items on the list.  SELECT fname, Iname, salary FROM patient WHERE salary NOT IN (10000, 20000, 30000) ; | Example 6.5b (Order ofprecendence)  Next, take a look at the order of logical operators. Because the WHERE clause can contain multiple types of operators, you need to understand the order in which they're resolved: t?] Arithmetic operations are solved first. Comparison operators are solved next. Logical operators have a lower precedence and are evaluated last in the order NOT, AND, and OR.  -- The AND clause is gotng to be done first and then the OR clause. This means it first  --100ks at the salary range and comes back with a result. If the salary is in the range or  -the City Davis then the record is selected.  SELECT Inarne, DOB, salary, city FROM patient WHERE scalar OR city='Davio t ; AND |
| **8.8 Dates and group functions**  **—Displays oldest person, youngest person, the number of records (excludes all those that have a NULL in --DOB), and number of records (Suppresses duplicate DOB).**  **SELECT min (DOB) , max (DOB) , count (DOB) , count (DISTINCT DOB) FROM patient;**  **--lnvalid: Cannot apply AVG to date formats. I-Ise months\_between to convert it into a number —and then do an average.**  **SELECT AVG (DOB) FROM patient;**  **--lnvalid: can do a sum on date formats.**  **SELECT SUM (DOB) FROM patient;** | **Example 6.6b (All)**  **Compares a value to every value in a list or returned by a query. Must be preceded by =, > , < , Evaluates to TRUE if the query returns no rows**  **--The salary has to be greater than both of those values. Once again is logically not correct.**  **SELECT fname, Inane, salary FROM patient WHERE salary > 30000 AND salary>40000;**  **--Same as above**  **SELECT fname, Iname, salar FROM atient WHERE salar >ALL (30000, 40000) ;** | **6.8 NULL**  **When performing arithmetic operations or search conditions, NULL values can cause unexpected results. A NULL value means no value has been stored in that field. Don't confuse a NULL value with a blank space. A NULL is the absence of data in a field; a field containing a blank space does contain a value— a blank space. When searching for NULL values, you can't use the equal sign because there's no value to use for comparison in the search condition. When checking for a NULL value, you're actually checking the status of the column: Does data exist or not? If you need to identify records that have a NULL value, you must use the IS NULL comparison operator.**  **INSERT INTO patient values (978, 'john' , 'Doe' , ' m', '11-FEB1978' , 25000, NULL, 'CA') ;**  **SELECT fname, 1 name,city FROM patient;**  **--Not correct. When checking for NULL , the (IS) keyword must be used.**  **SELECT fname, Iname,city FROM patient WHERE city= NULL;** |
| 6.9 Creating tables with select statements  Example 6.9a (Creating table using select)  --Can feed the results to another table instead of displaying it to the screen. The new —table (patient\_temp) will be comprised of fname, Iname, salary. This new table will --contain the data from the patient table.  --Keep in mind that the column names must be valid identifiers, which means that things —such as calculations or function names which have parentheses in them will not be  --accepted. For this reason, an alias is used. SELECT fname, cit from atient;  CREATE TABLE patient temp AS SELECT fname, Iname, salary, city FROM patient WHERE city— 'Mavis';  SELECT \* FROM patient temp; | E.vmnple 6.9b (Creating tables and aliases)  --Creating a table and using an alias name  'TABLE patient \_ temp;  —INVALID: have to use an alias.  CREATE TABLE patient temp SELECT fname, Inane, salary\*2 FROM patient WHERE salary >30000;-  CREATE TABLE patient temp AS SELECT fname, Inarne, sa lary\*2 new salary FROM patient WHERE salary >30000;  —Notice the column name new\_salary.  DESC patient tem | **Example 6.9c (Creating table with constraints)**  **In creating a copy, only the not NULL constraint gets copied over. Also the name of the constraint will be different for the new table**  **DPOP TABLE test;**  **DROP TABLE test2;**  **COLUMN search condition FOPMAT AIO**  **CREATE TABLE**  **coi I NUMBER PP. IMAPY K FTI, co] 2 UUMBEP (JUIOUF„, col 3 UUMBEP CHECK (co] 3>20) ,**  **NUMBER NOT NULL** |
| **6.10 Updating tables using the update statement**  **The UPDATE statement allows you to update a single record or multiple records in a table.**  **The syntax for the UPDATE statement is:**  **UPDATE table SET column = expression WHERE predicates;**  **143**  **Example 6.10a (Updating)**  **SELECT fname, 1 name, salary FROM patient where patient id—l Il;**  **—Updates the patient table by setting the fname, Iname and salary for patient (11). UPDATE patient SET fname-'Bi11' Iname=' Bob' , salary—**  **Example 7c (Secondary sorts)**  **In the previous examples, only one column was specified in the ORDER BY clause, which is called a primary sort. In some cases, you might want to include a secondary sort, which specifies a second field to sort by if an exact match occurs between two or more rows in the primary sort.**  **149**  **--0rder% by fname, When it comes across, fnames that are the same, then it further sorts •by Iname. Both sorts are in ascending order.**  **SELECT patient id, fname, Iname,city CITYNAME FROM patient ORDER BY fname, 1 name ;**  **--Can make fname in descending and Iname in ascending order which is default.**  **SELECT patient\_id, fname, 1 name, city CITYNAME FROM patient ORDER BY fname DESC 1 name ;** | **Example 7a (Order by)**  **The ORDER BY clause, used to display query results in a sorted order, is listed at the end of the SELECT statement,The columns used to sort the results are listed in the ORDER BY clause.ln the query results, the second column ( Name) is listed in ascending alphabetical order. Note these important points:**  **--Default sort is in ascending order.**  **SELECT patient id, fname, Iname, city CITYNAME FROM patient ORDER BY city;**  **--ASC is implied by default.**  **Example 7d (Position)**  **Oracle also provides an abbreviated method for referencing the sort column if the name is used in the SELECT clause. In the previous example, State and City are used in both the SELECT and ORDER BY clauses. Instead of listing these column names again in the ORDER BY clause, you can reference them by their positions in the SELECT clause's column list. You can also use the column alias.**  **--Can use the order by on the actual column, the position of where it appears between —the select and from or by the alias.**  **SELECT patient id, fname, Inarne, city CITYNAME FROM patient ORDER BY 4;**  **SELECT patient\_id, fname, 1 name, city CITYNAME FROM patient ORDER BY CITYNAME;** | Example 7b (NULLs)  When sorting in ascending order, values are listed in this order:  1. Blank and special characters  2. Numeric values  3. Character values ( uppercase first)  4. NULL values  Unless you specify " DESC" for descending, the ORDER BY clause sorts in ascending order by default.lf a column alias is given to a field in the SELECT clause, you can reference the field in the ORDER BY clause with the column alias— although doing so isn't required.You can also use the ORDER BY clause with the optional NULLS FIRST or NULLS LAST keywords to change the order for listing NULL values. By default, NULL values are listed last when results are sorted in ascending order and first when they're sorted in descending order.  --NULLs appear last by default. You can change the default to make them appear first.  SELECT patient id, fname, Iname, city CT TYNAME FROM patient ORDER BY city  NULLS EIR5T;  --NULLs appear last which is implied.  SELECT patient id, fname, Iname, city CITYNÄME FROM patient ORDER BY city NULLS LAST; |
| **8.2 SUM**  **The SUM function is used to calculate the total amount stored in a numeric field for a group of records. The syntax of the SUM function is SUM(([ DISTINCTI ALL] n), where n is a column containing numeric data.**  **SELECT salary from patient;**  **—The result is a single number that is the summation of all the salaries. SELECT SUM (salary) FROM patient;**  **--AII is implied as in the above statement.**  **SELECT SUM (ALL salary) FROM patient;**  **--The result is a single number that is the summation of all the distinct salaries, which --means it suppresses the duplicates** | **before doing a summation.**  **SELECT SUM (Dls rrmcrr salary) FROM patient;**  **Creates a grouping for each city, which means that it suppresses the duplicates in each group and --and then comes up with a summation for each group order by is the last clause.**  **SELECT city, SUM (salary) FROM patient GROUP BY city ORDER BY 1;**  **--Creates a group for each of the states and gives a summation for each state.**  **SELECT state, SUM (salary) FROM patient GROUP BY state;**  **--Creates combination of fname, city catagories and provides a summation for each group.** | **SELECT fname, cit SUM (salar ) FROM patient GROUP BY fname**  **--First the where clause filters. Then it does a grouping with the data that is left over. It**  **--groups the different cities and then for each city group, it comes up with a summation.**  **SELECT city, SUM (salary) FROM patient WHERE UPPER 'RENO' BY city ORDER BY 1; GROUP**  **--First the where clause filters. Then it does a grouping with the data that is left over. It**  **—groups the different cities and then for each city group, it comes up with a summation.**  **SELECT city, SUM (salary) FROM patient WHERE UPPER (city) 'RENO' or city is NULL GROUP BY cit ORDER BY 1;** |
| **8.4 AVG**  **The AVG function calculates the average of numeric values in a specified column. The syn-tax of the AVG function is AVG([ DISTINCTI ALL] n), where n is a column containing numeric data.**  **SELECT salary FROM patient;**  **--Gives a single average for all the salaries.**  **SELECT AVG (salary) FROM patient;**  **—Same as above**  **SELECT AVG (ALL salary) FROM patient;**  **—Suppresses duplicates and then gives an average.**  **SELECT AVG (DISTINCT salary) FROM patient;**  **--lnvalid: Does not know how to display the one single average salary with all the cities. SELECT city, AVG (salary) FROM patient;**  **--Displays the average salary for each city category.**  **SELECT city, AVG (salary) FROM patient GROUP BY city ORDER BY 1;**  **--First it filters the data based on the where clause. Then it takes the left over records and --does a grouping for each of the cities and provides an average for each grouping.**  **SELECT city, AVG (salary) FROM patient WHERE UPPER (city) <> 'RENO' GROUP BY city ORDER BY 1;**  **—First it filters the data based on the where clause. Then it takes the left over records and**  **--does a grouping for each of the cities and provides an average for each grouping. If there is a**  **--city group that does not have a salary, which means that it is NULL, then it will replace it with a**  **--zero. There is additional filtering using the having clause after all the grouping is done,**  **SELECT city, AVG (nvl (salary, 0) ) FROM patient WHERE UPPER (city) <> 'RENO'**  **GROUP BY city HAVING AVG (salary) >20000 ORDER BY 1;** | **8.5 COUNT**  **Depending on the argument used, the COUNT function can count the records having non- NULL values in a specified field or count the total records meeting a specific condition, including those containing NULL values. The syntax of the COUNT function is COUNT(\* [ DISTINCTI ALL] c), where c represents a numeric or non-numeric column.**  **SELECT fname, 1 name, city FROM patient;**  **--Counts the number of rows.**  **SELECT COUNT (+) FROM patient;**  **--Counts the number rows based on the contents of the city. If the city for a given row contains --a NULL, then it will not be counted.**  **SELECT CO(JWr (city) FROM patient;**  **--Same as above**  **SELECT COUNT (ALL city) FROM patient;**  **--lnvalid: Does not know how to display a single number with the six different cities.**  **SELECT city, COUNT ( \* ) FROM patient;**  **--Create a group for each of the different cities and do a count for each category. NULL cities are —excluded from the count.**  **SELECT city, COUNT (city) FROM patient GROUP BY city;**  **—Same as above but the NULLs are not excluded.**  **SELECT city, COUNT ( w ) FROM patient GROUP BY city;**  **--After it has come up with the count per grouping, there is an additional filtering, which includes only those --records where the count is greater than I.**  **SELECT city, COUNT FROM patient GROUP. BY city HAVING COUNT** | **8.6 MAX**  **The MAX function returns the largest value stored in the specified column. The syntax of the MAX function is MAX([ DISTINCTI ALL] c), where c can represent any numeric, character, or date column.**  **SELECT salary FROM patient;**  **--The highest salary is displayed.**  **SELECT MAY. (salary) FROM patient; --same as above**  **SELECT MAX (ALL salary) FROM patient;**  **--lnvalid• the highest salary is a single number and cannot be associated with all cities. SELECT city, MAX (salary) FROM patient;**  **--Displavs highest salary for each city.**  **SELECT city, MAX (salary) FROM patient GROUP BY city;**  **—Given the fname, city combination, display the the number of records and the highest salary —for each of those combination categories.**  **SELECT fname, city, AVG (salary), MAX (salary) FROM patient GROUP BY fname, city;**  **—Same as above except that after the final result, do some additional filtering based on the count.**  **SELECT fname, city, AVG (salary) , MAX (salary) FROM patient GROUP BY fname, city HAVING COUNT (+) > 2;** |
| **8.7 MIN**  **In contrast to the MAX function, the MIN function returns the smallest value in a specified column. As with the MAX function, the MIN function works with any numeric, character, or date column. The syntax of the MIN function is MIN([ DISTINCTI ALLJ c), where c represents any character, numeric, or date column. The MIN function uses the same logic as the MAX function for numeric and character data, except it returns the smallest value rather than the largest value.**  **SELECT salary FROM patient;**  **--The lowest salary is displayed.**  **SELECT MIN (salary) FROM patient;**  **—Invalid: cannot display a single number with six cities.**  **SELECT city, MIN (salary) FROM patient;**  **—Display the lowest salary for each city category and display the number of records in each group.**  **SELECT city, MIN (salary) , COUNT (O FROM patient GROUP BY cit**  **-Jilters with the where clause. Then given the remaining records, it groups by city and finds the lowest salary lor each city category. Given the result set, it only includes the ones where -there are more than two records for each group. The results are sorted by city.**  **SELECT city, MIN (salary) FROM patient WHERE city IS NOT NULL GROUP BY city HAVING ORDER BY 1 DESC;** | **10000 WHERE patient id=lll;**  **--Confirm results.**  **SELECT fname, I name, salary FROM patient where patient id=lll;**  **8.1 Group by examples**  **SELECT SUM (salary) FROM patient;**  **SELECT cit SUM (salar ) FROM atient GROUP BY city;**  **SELECT state, SUM(sa1ary) FROM patient GROUP BY state;**  **SELECT gender, SUM (salar ) FROM patient GROUP BY gender;**  **ELECT city, gender, SUM (salary) FROM patient CROUP BY cit , gender;**  **SELECT state, gender, SUM (salary) FROM patient GROUP BY state, gender;**  **SELECT fname, city, SUM(sa1ary) FROM patient GROUP BY fname, city;** | **SELECT patient id, fname, Iname, city CITYNAME FROM patient ORDER BY city ASC; --Can sort in descending order.**  **SELECT patient id, fname, Iname, city CITYNAME FROM patient ORDER BY city DESC;**  **--Can sort by the position 01 the column between the select and from clause.**  **SELECT patient \_ 1.1, Jname, city CJTYNAME, FROM patient ORDER BY 4;**  **—Can use the alias name for sorting.**  **SELECT CITYNAME FROM patient ORDER BY CITYNAME;**  **Example 7e (column versus alias)**  **Difference in ordering by the column versus by the alias**  **--1n this dataset there is one record that has a NULL for salary.**  **--0rder by alias after zero has been substituted.**  **SELECT WI. (salary, O) pay FROM patient ORDER BY pay;**  **--0rder by position after zero has been substituted.**  **SELECT (salary, O) FROM patient ORDER BY 1;** |

**Quizzes**

1. Which query will list the customer living in the same state as the customer named leila smith. Note:There are several people with the name smith.

1. select customer from customers where state=(select state from customers where lastname =’smith’);
2. select customer from customers where state= (select state from customers where lastname =’smith’ and firstname=’leila’)
3. select customer from customers where state=(select state from customers where lastname =’smith’ or firstname=’leila’ )
4. none of the above

2) Which of the following is valid considering that there are several records in the orders table with order number 1010.

1. select order from orders where shipdate=select shipdate from orders where order=1010
2. select order from orders where shipdate=(select shipdate from orders) and order=1010
3. select order from orders where shipdate=(select shipdate from orders where order=1010)
4. select order from orders having shipdate=(select shipdate from where order=1010)
5. none of the above

3) Which of the following is considered a multiple  row operator

1. IN             b)   =     c) <>   d) All the above
2. none of the above

4) What is the purpose of the following query

select isbn, title from books where (pubid, category) in

(select pubid, category from books where title like ‘%oracle%’);

1. it lists all the publishers and categories containing the value oracle
2. it lists the isbn and title of all books belonging to the (same category and having the same publisher) as any book with the letters oracle in its title
3. All the above
4. None of the above

5) Given the following query considering that there is one order# with isbn ’99-99-99’ in the orderitems table and that order# appears once in in the orders table.

select order from orders where order in (select order# from orderitems where isbn=’99-99-99’)

1. the statement will not execute since the subquery and the outer query do not reference the same table
2. the outer query is not necessary since it has no effect on the results displayed
3. the query will fail if only one result is returned to the outer query since the where clause of the outer query uses the In comparison operator
4. No rows will be displayed since the isbn in the where clause is enclosed in single quotation marks
5. None of the above

6)  A complete query nested inside another query is called a(n) \_\_\_\_.

|  |  |  |  |
| --- | --- | --- | --- |
| a. | inner view | c. | child view |
| b. | Subquery | d. | all of the above |

      7) Which of the following must be used to separate a subquery from the outer query?

|  |  |  |  |
| --- | --- | --- | --- |
| a. | / / | c. | | | |
| b. | " " | d. | ( ) |

      8) The <> operator is referred to as a(n) \_\_\_\_ operator.

|  |  |  |  |
| --- | --- | --- | --- |
| a. | single-row | c. | multiple-column |
| b. | multiple row | d. | Correlated |

     9) If it is possible for a subquery to return a NULL value to the outer query for comparison, the \_\_\_\_ function should be used to substitute an actual value for the NULL.

|  |  |  |  |
| --- | --- | --- | --- |
| a. | IS NULL | c. | !NVL |
| b. | NVL | d. | NVLS |

1) In which clause should you include the condition MAX (cost) > 39 to restrict the groups displayed in the results of a query?

a) group by            b) where  c) having            d) select e) none of the above

2) Which group function returns the lowest numeric or character value?

a)least   b)minimum             c)min d)small             e)smallest

3) Which of the following is valid

a) SELECT category, avg (cost) FROM books GROUP BY cost;

b) SELECT category, avg (cost) profit FROM books GROUP BY category  HAVING profit > 5;

c) SELECT category, avg (cost) FROM books GROUP BY category;

d) none of the above

4)  How would you display a listing of the sums of employee salaries for those employees not making a  commission, for each job type, including only those sums greater than 2500?

  a) SELECT job, sum(sal) FROM emp  WHERE sum(sal) > 2500 AND comm IS NULL;

  b) SELECT job, sum(sal)  FROM emp WHERE comm IS null   GROUP BY job HAVING sum(sal) > 2500;

                c) SELECT job, sum(sal) FROM emp WHERE sum(sal) > 2500 AND comm IS NULL GROUP BY job;

  d) none of the above

* **For aggregate functions (COUNT, MAX, MIN, SUM, AVG), GROUP BY clause is mandatory.**

5).  The following SQL statement is illegal because: (choose one)

**SELECT deptno, AVG(sal)    FROM emp WHERE AVG(sal)> 2000    GROUP BY deptno;**

a. It requires data from more than one table, yet only one table is listed.

b. sal is not a legal column name.

c. Instead of a WHERE clause, a HAVING clause must be used to restrict groups.

d. The GROUP BY clause must contain AVG(sal).

e. This SELECT statement is perfectly legal.

6) Which of the following choices will result in an error?

a) SELECT deptno, job, sum(sal) FROM emp GROUP BY job, deptno;   
b) SELECT sum(sal), deptno, job FROM emp GROUP BY job, deptno;   
c) SELECT deptno, job, sum(sal) FROM emp;   
d) SELECT deptno, sum(sal), job FROM emp GROUP BY job, deptno;

7) Which one gives you the number of animals per zoo?

a) SELECT zoo, COUNT(animal) FROM AnimalZoo GROUP BY COUNT(animal);

b) SELECT COUNT(zoo), count(animal) FROM AnimalZoo GROUP BY COUNT(zoo);

c) SELECT zoo, SUM(animal) FROM AnimalZoo GROUP BY SUM(animal);

d) SELECT SUM(zoo), animal FROM AnimalZoo GROUP BY SUM(zoo);  
e) NONE of the above

8)  Which clause gets executed last in an SQL statement?

a) GROUP BY

b) HAVING

c) ORDER BY

d) FROM

e) SELECT

1) How many rows will be returned in a Cartesian product of two tables one which has 5 records and the other 10?

Answer: 50. Its simply m\*n

2) What is CROSS JOIN?

Answer: CROSS JOIN provides combination of every row from two tables. If two tables have ‘m’ and ‘n’ rows each then result contains m\*n rows.

3) How many join conditions do you need in the where clause to do an inner join between five tables?

Answer: 4. Its n-1 conditions, where ‘n’ is number of tables

4) Which of the following create a cartesian product

a) SELECT title, authorid FROM books, bookauthor

b) SELECT title, name FROM books , bookauthor WHERE

books.isbn=books.isbn

c) SELECT \* FROM Books CROSS JOIN BookAuthor

d) all the above

e) a and b

f) a and c

g) b and c

Answer: d

**These tables pertain to questions 5-9**

|  |  |
| --- | --- |
| **Candidate**  name id  ------- ---  Jack 1  John 2  Jill 1  Joe 9 | **Party**  id description  -- ---------------------  1 republican  2 democrat  3 independent  4 Green  5 libertarian |

5) How do you get the following result set? (Write the SQL Statement)

name description

------ --------------

Jack republican

john democrat

jill republican

Answer: select c.name, p.description from candidate c, part p wherec.id=p.id;

|  |  |
| --- | --- |
| 6) How do you get the following result set? (Write the SQL Statement)    name description  ------ --------------  Jack republican  john democrat  jill republican  Independent  Green  Liberterian | 7) How do you get the following result set? (Write the SQL Statement)    name description  ------ --------------  Jack republican  john democrat  jill republican  joe |

8) How do you display the id that is in the party table but not in the candidate table using exists, in and minus? (Write the SQL Statement)

9) How do you get a unique listing of all the ids that are in both tables (No duplicates)? (Write the SQL Statement)

Select id from table1 union select id from table2;

10) which of the following is not valid. (b)

1. select isbn from books

minus

select isbn from orderitems

1. select isbn from books, publisher where books.pubid(+)=publisher.pub\_id(+)
2. none of the above are valid
3. all of the above are valid

12) which of the following statements regarding an outer join between two tables is true? (b)

1. if the relationship between the tables is established through the where clause then both tables can include the outer join operator
2. To include unmatched records in the results, the record is paired with a Null record in the deficient table
3. none of the above, d) all the above

**Group Order by**

|  |
| --- |
| 1) Display the lastname, firstname and salary of everyone in the candidate table. Order in ascending order based on lastname. Use the column name to sor |
|  |
| 2) Display the lastname, firstname and salary of everyone in the candidate table. Order in Descending order based on the combination of lastname and first name Do not use the column names but rather their position |
|  |
| 3) Display the lastname, firstname and salary \* 2 (use alias double\_salary) of everyone in the candidate table. Order in Descending order based on the alias |
|  |

**Group Subquery**

|  |
| --- |
| 1) Display the first name of all the candidates who are Democrats. (use subquery) |
|  |
| 2) Display the party description of the all the candidates whose last name ends with ‘C’ regardless of case. (Use subquery) |
|  |
| 3) Display the first name of all the candidates who are either Democrats or Republicans regardless of case. (use subquery, use in clause) |
|  |
| 4) Create a second table called candidate2 that contains all the data from the candidate table using create table as… statement. The new table should contain only the first name, lastname and the salary, lowered by 10% for all the candidates who are Republicans |
|  |

**Group Group by**

|  |
| --- |
| 1) Display the number of people in each party. Order by Party. (Make sure to display the party\_code) |
|  |
| 2) Display the number of people in each party whose first name starts with d or r. (Make sure to display the party\_code) |
|  |
| 3) Display the average salary for each party (Make sure to display the party\_code) |
|  |
| 4) Display the number of people in each party where the number of people does not exceed 2 |
|  |
| 5) Display the average salary for each party where the average does not exceed 50000 |
|  |
| 6) Create a new table called candidate2 that contains the number of people in each party. Should contain the partycode and the number of people (CAUTION, you have to use an alias for this to work) |
|  |

**Group Inner join**

|  |
| --- |
| 1) Display the Cartesian product of the party and candidate table |
|  |
| 2) Display the lastname and the party description of each individual |
|  |
| 3) Display the last name and the party dscription of each individual. If there is not a party description, then display no description. (Use the NVL function) |
|  |
| 4) Display the number of people in each party (display party\_description) |
|  |
| 5) Display the number of people in each party for only those parties whose average salary is greater than 50,000. (Identify the party name) |
|  |

**Group Outer Join**

|  |
| --- |
| 1) Display the last name and the party description of each individual. If there is not a party associated with the individual then display “no party” |
|  |
| 2) Display the last name and the party description of each individual. If there is a party that does not have any individuals associated with it, then display “No one” |
|  |

**Group Union**

|  |
| --- |
| 1. Display the name and the corresponding description based on the candidates salary **(Use union and be cautious of the types) Order by name**   < 50000 Honest  >50000 and < 100000 Not so honest  >100000 80000 |
|  |
| 1. Display the name and the corresponding description based on the candidates salary (**Use a plain case statement in chapter 5)**   < 50000 Honest  >50000 and < 100000 Not so honest  >100000 80000 |
|  |
| 3) Display the name of all the people who are not associated with a party (use not in). This is a bit tricky because people who don’t have a party\_id will have a null and you cannot compare a null using not in because it doesn’t know how to deal with non-data. You can put the party\_id in an NVL function in the where clause to resolve this issue |
|  |
| 4) Repeat question 3 using (not exists) |
|  |
| 5) Repeat question 3 using (minus) |
|  |