Advanced sql queries examples with answers pdf

These questions are most often asked in interviews. To get alternate entries from the table. (EVEN NUMBERED) choose from emp, where rowid in (choose decod (mod (mod (rownum,2),0,rowid, null) from emp); Select ALTERNATE entries from the table. (ODD NUMBERED) choose from the emp, where the rowid is in (choose the decod (mod (rownum,2),0,null, rowid) from emp); Find the 3rd MAX salary in the emp table. Choose different fats from emp e1, where 3 (choose a stake (different fat) from emp e2, where e1.sal zlt; e2.sal); Find the 3rd minute salary in the emp table. Choose different fats from emp e1, where 3 (choose a stake (different fat) from emp e2where e1.sal qgt; e2.sal); Select FIRST n entries from the table. Choose from the emp, where rownum is; 'n; Choose the latest n entries from the table select - from emp minus select - from emp, where rownum zlt; (choose count)- n from emp); Department No. list, Department Name for all departments that don't have employees in the department. Choose from a department where deptno is not in (choose deptno from emp); Alternative solution: Select q from Division A, where it is not present (choose from emp b, where a.deptno and b.deptno); modified solution: select empno,ename, b.deptno, dname from emp a, dept b, where a.deptno (yap.) - b.deptno and empno are invalid; How to get a 3 Max salary? Choose a variety of fats from emp, where 3 zgt; (choose a stake (different sal) from emp b, where a.sal zlt; b.sal) order a.sal desc; How to get a 3 min salary? Choose a variety of fats from emp, where 3 zgt; (choose a stake (different fat) from emp b, where a.sal zgt; b.sal); How to get nth maximum salary? Choose a different hireate from emp, where n (choose a stake (different sal) from emp b, where a.sal zgt; b.sal); Choose DISTINCT RECORDS from the emp table. choose from emp A, where rowid (choose max (rowid) from emp b, where a.empno'b.empno); How do I remove string duplicates from a table? remove from emp a where rowid!' (choose max (rowid) from emp b, where a.empno'b.empno); Count the number of employees in department wise. select count (EMPNO), b.deptno, dname from emp a, dept b where a.deptno (i) group b.deptno,dname; Suppose there is an annual wage information provided by the emp table. How do I get a monthly salary for each employee? Choose ename,sal/12 as a monthly from emp; Choose all entries from the emp table where deptno No 10 or 40. Choose from an emp where deptno-30 or deptno-10; Select all entries from the emp table, where deptno-30 and sal'gt;1500. Choose from an emp where deptno-30 and sal'gt;1500; Choose all entries from emp where work is not in SALESMAN or CLERK. Choose from an emp where the work is not in ('SALESMAN', 'CLERK'); Select all entries from emp, where the electronic name is in 'BLAKE', 'SCOTT', 'KING'and'FORD'. Choose from the emp, where the electronic name is in ('JONES', 'BLAKE',' 'SCOTT', 'KING','FORD'); Choose all entries where the electronic name starts with 'S' and its lenth is 6 char. Choose from an emp where ename like'S____'; Choose everything где электронное имя может быть какой-либо нет характера, но она должна закончиться с 'R'. выбрать из emp, где электронное имя, как'%R'; Граф MGR и их зарплата в таблице emp. выбрать кол (MGR), кол (sal) от emp; В таблице emp добавить comm'sal как общий сал . выбрать электронное имя,(sal'nvl (comm,0)) в качестве итога от emp; Выберите любую <3000 from= emp= table. = select= *= from= emp = where= sal=>зарплату any (выберите sal от emp где sal <3000); select = all= salary=></3000);> <3000 from= emp= table. = select= *= from= emp = where= sal=> все (выберите sal от emp где sal<3000); select= all= the= employee = group= by= deptno= and= sal= in= descending= order.= select= ename,deptno,sal= from= emp= order= by= deptno,sal= desc;= how= can= i= create= an= empty= table= emp1= with= same= structure= as= emp?= create= table= emp1= as= select= *= from= emp= where= 1=2; how= to= retrive= record= where= sal= between= 1000= to= 2000?= select= *= from= emp= where= sal=>1000 и sal<2000 select= all= records= where= dept= no= of= both= emp= and= dept= table= matches.= select= *= from= emp= where= exists(select= *= from= dept= where= emp.deptno=dept.deptno) if= there= are= two= tables= emp1= and= emp2,= and= both= have= common= record.= how= can= i= fetch= all= the= recods= but= common= records= only= once?= (select= *= from= emp)= union= (select= *= from= emp1)= how= to= fetch= only= common= records= from= two= tables= emp= and= emp1?= (select= *= from= emp)= intersect= (select= *= from= emp1)= how= can= i= retrive= all= records= of= emp1= those= should= not= present= in= emp2?= (select= *= from= emp)= minus= (select= *= from= emp1)= count= the= totalsa = deptno= wise= where= more= than= 2= employees= exist.= select = deptno ,= sum(sal)= as= totalsal= from= emp= group= by= deptno= having= count(empno)=> 2 Страница 2 Страница 1 из 3 Запрос SELECT самый часто используемый запрос в любой базе данных. Таким образом, изучение этого запроса очень важно. Даже если вы не пишете его вручную, так как некоторые инструменты облегчают задачу, предоставляя код помогает. Но этот запрос поможет вам понять многие аспекты концепций баз данных, начиная от задачи DBA и до PL/S'L. Я надеюсь, что этот учебник помогает узнать SELECT запрос быстрее и проще. Мы будем ссылаться таблицы из схемы по умолчанию / пользователь SCOTT в Oracle. Обобщенный выбранный запрос выглядит так: SELECT BCE, что я не хею. «ColumnName » AS newColumnName , «columnName1 » AS newColumnName1», ...,......,...,...,...,...,...,...,...,...,...,..., «OT tableName <condition>»ALIAS», «ГДЕ» (GROUP BY <condition>columnlist) Выберите все детали сотрудника из таблицы emp. SELECT - FROM emp; Выберите только имена всех сотрудников в таблице emp; SELECT электронное имя от emp; Выберите записи с помощью нескольких столбцов; SELECT eno,ename, sal FROM emp; Выберите записи, псевдоним; SELECT eno,ename, зарплата AS sal FROM emp; Выберите записи, используя несколько псевдонимов SELECT eno AS empno, ename AS empname, sal AS salary FROM emp; Использование функций для усиления псевдонима SELECT count (sal) в качестве totalsal FROM emp; Производные или</condition> </condition> </2000> </3000);> </3000> </3000> </3000> Fields: Column values were manipulated as repeated. Find a monthly employee's salary, (salary is kept on the basis of Annum). SELECT sal / 12 FROM emp; Use Alias to decorate the fields you receive or calculate. SELECT sal / 12 AS monthly_salary from emp Calculate the amount of monthly salary and commissions of the employee. SELECT electronic name, (sal / 12) Nvl (comm.0) AS monthsalwithcomm from emp; S'L is incredibly powerful, and like any well-made development tool, it has several commands that are vital for a good developer to know. Here's a list of S'L queries that are really important for coding and optimization. Each of the queries in our SL tutorial has value for almost every system that interacts with the S'L database. 1. Requesting tables to receive tables can be launched to obtain a list of tables present in the database where the database My_Schema. With select command, users can identify the columns they want in the exit of the query. This command is also useful for finding out which column users want to see as a output table. The SELECT statement is used to collect data from the table. The resulting data will be included in the results table, called the set of results. The output is stored in the results table. This output table is also called a set of results. CHOOSE FROM MY_SCHEMA. Tables; 2. Request to select columns from the table This is perhaps the most widely used example of S'L queries. In the example below, we remove a Student_ID column or attribute from the STUDENT table. Select an extract to select data from the database. PICK STUDENT_ID FROM A STUDENT; If you want to display all the attributes from a particular table, this is the right query to use: SELECT - FROM STUDENT; 3. Request for data output by restriction This S'L request extracts these attributes from the table on the employee ID of the no0000 SELECT EMP_ID, NAME FROM EMPLOYEE_TBL WHERE EMP_ID and '0000'; 4. Request for the output of sorted data using 'Order By' This query orders results in relation to the attribute to which Order By refers - for example, if this attribute is a more integral type of data, the result will either be sorted in ascending or descending order; similarly, if the data type is a line, the result will be ordered in alphabetical order. The reservation order is used to sort the data out of the table. The reservation order should always be used in the last S'L request. SELECT EMP_ID, LAST_NAME FROM SOUTH EMP_ID The result order can also be set manually, using asc for ascending and desc for descent. Climbing (ASC) is a default condition for ORDER BY. In other words, if users do not specify ASC or DESC after the column name, the result will only be ordered in ascending order. Increasing. EMP_ID, LAST_NAME from EMPLOYEE_TBL WHERE CITY - INDIANAPOLIS ORDER BY EMP_ID asc; 5. Requesting the output of sorted data using 'Group By' The 'Group By' groups the data in accordance with the specified attribute. The S'L query below will select Name, Age columns from the Patient Table, and then they will be filtered by age to include entries where the age is over 40 years old, then group entries with a similar age value, and then finally turn them off by name. The basic rule is that the reservation team should always follow where the item in the Select statement is, and should precede the Order of the item. PICK NAME, AGE FROM patients, 40 YEARS ON NAME, AGE ON NAME; Another example of using Group By: this expression will select entries with a price of less than 70 from the Order table, will be a group of entries with a similar price, will sort the output by price, and add a column OF COUNT (price) that will display how many entries with a similar price have been found: SELECT COUNT (price), price from orders where the price is lt; 70 GROUP BY PRICE ORDER Price Note Note A: You have to use the same set of columns for both SELECT and GROUP BY commands, otherwise you will get an error. Thank you so much Sachidannad for pointing out! Requests to manipulate data using mathematical functions there are many built-in mathematical functions such as COUNT and AVG that provide basic functionality of counting the number of results and averaging them accordingly. 6. Manipulating data with COUNT This query shows the total number of customers counting each customer's ID. In addition, it groups results depending on the country of each customer. In the graph, if users define DISTINCT, then they cal also identify query_partition_clause. This provision is part of the analytical situation, and other provisions, such as order_by_clause and windowing_clause are not allowed. Syntax: SELECT COUNT (kolnam) from the table name; SELECT COUNT (CustomerID), Country from GROUP BY Country customers; 7. Manipulating data using SUM SUM calculates the total amount of attribute that is given to it as an argument. SUM is an aggregated function and calculates

the amount of all different values. and the sum of all duplicate values. SELECT SUM (salary) FROM the employee, WHERE Emp_Age qlt; 30; 8. Manipulating data using AVG Simple is the average of this attribute. The average is also an aggregated function in S'L. AVG calculates the average non-NULL in the column. It ignores zero values. PLEASE AVG (Price) FROM products; 9. Request for a list of all submissions This S'L query lists all the submissions available in the scheme. PICK FROM My_Schema.views; 10. Requesting a view is a table that was formed as a result of the request. It has tables and strings like any other table. It is usually a good idea to run S'L queries as an independent independent because it allows them to be extracted later to view the results of the query rather than calculating the same command each time for a specific set of results. THE FAILING_STUDENTS HOW TO CHOOSE S_NAME, STUDENT_ID FROM A student, WHERE GPA IS zgt; 40; The standard syntax for the selection of attributes from the table also applies to views. PICK FROM FAILING_STUDENTS; 12. Requesting an update to this request updates a submission called The Product List - and if that view does not exist, the product list is submitted as stated in that query. The view is also called a virtual table. In other words, the view is only a mirror image of a table whose data is the result of a saved query. The view is a legitimate copy of another table or sequence of tables. The view receives information or data from tables from previously created tables, known as base tables. Base tables are real tables. All the procedures implemented on the view do change the base table. Users can use views in the same way as real or base tables. In the view, users can use different DDL, DML commands, such as upgrades, insertions, and deletions. CREATE OR NAME (Product List) HOW TO READ ProductID, ProductName, ProductName Category, WHERE discontinued - No; 13. Request to delete submission This query will drop or delete a view called 'V1'. It is important to remember that DROP VIEW is prohibited if there are any opinions that depend on the kind you are about to fall. DROP VIEW V1; 14. Requesting user tables The table, determined by the user, is a representation of certain information in the table and can be used as an argument for user-defined procedures or functions. Because they're so useful, it's helpful to keep track of them with the next query. User tables explain the current user's relational tables. SELECT - FROM Sys.objects WHERE Type'u' 15. The Key Key request clearly defines all the values in the table. The main key imposes a restriction NOT NULL and a unique limitation in one declaration. In other words, it prevents similar values or column sequences. It doesn't allow zero values. The primary key can be defined on one column or a combination of two columns in the table. He is responsible for all the relationships between the tables. The following S'L query lists all the fields in the main table key. SELECT - from Sys.Objects WHERE Type 'PK' 16. Requesting a display of unique keys Unique Key allows the column to make sure that all its values are different. The unique key also recognizes bundle is unique in relation to or table. The table can have more than one unique key. Unique key limitations can only take one NULL value for a column. SELECT - FROM Sys.Objects WHERE Type'uq' 17. Displaying foreign keys Foreign keys link one table to another - - attributes in one table that belong to the main key of another table. SELECT - FROM Sys.Objects WHERE Initial, Unique and Foreign Type'f are part of the limitations in S'L. Restrictions are important for scalability, consistency, and data sincerity. Restrictions implement specific rules to ensure that these are compliant with the conditions set out in them. For example, these are laws that are imposed on the columns of database tables. They are used to limit the type of data in the table. This ensures that the database is effective and authentic. 18. Trigger triggers display is a kind of listener event - i.e., it is a predetermined set of instructions that are executed when a particular event occurs. You can view the list of specific triggers with the following query. SELECT - FROM Sys.Objects WHERE Type'tr' 19. Displaying internal tables Internal tables are formed as a user's snr of action and are usually unavailable. Data in internal tables cannot be used; however, the metadata of internal tables can be viewed with the following query. SELECT - FROM Sys.Objects WHERE Type 'is 20. Showing a list of procedures stored in the Storage Procedure is a group of extended S'L queries that logically form a single unit and perform a specific task. So using the following query, you can track them: SELECT and from Sys.Objects WHERE Type'p' Improving Your Skills with Coach S'L (for Beginners and Advanced Developers) Try S'L Trainer for free. and two more advanced S'L queries for our users! 21. By changing the values of the two columns in table In this and subsequent examples, we will use a common company database, including several tables that are easily visualized. Our DB practice will include a customer table and an order table. The Customer Table will contain some obvious columns, including ID, Name, Address, zip, and email, for example, where we assume that the key field for indexing is Customer_ID field. To this end, we can easily imagine an order table that also contains an indexed customer identification area, as well as detailed information about each order placed by the customer. This table will include the order number, number, date, item and price. In our first example, S'L imagine a situation where the lightning and phone fields were moved and all phone numbers were mistakenly entered into the postcode field. We can easily solve this problem with the following S'L statement: UPDATE Set-Phone Customers, Phone-Sip 22. Returning the unique value column now, let's assume that our data entry operator has mistakenly added the same Customer to the Customer table. As you know, proper indexing requires that the key field contains only Values. To solve this problem, we will use SELECT DISTINCT to create an indexed list of unique customers: SELECT DISTINCT ID From Customers Customers By doing top 25 with SELECT TOP Clause Next, imagine that our customer table has grown to include thousands of entries, but we just want to show a sample of 25 of these entries to demonstrate column headlines and the SELECT TOP position allows us to specify the number of entries to return as the top 25 list. In this example, we'll refund the top 25 of our customer table: SELECT TOP 25 from customers where Customer_ID.null; 24. Finding SDL tables with Wildcard characters or operators such as % makes it easy to find certain lines in a large table of thousands of entries. Suppose we want to find all of our customers who have names ranging from Herb, including Herberts, and Herbertson. You can use a wildcard symbol to achieve this result. The next S'L query will return all lines from the Customer table, where the Customer_name box starts with Herb: SELECT - From customers where the name LIKE 'Herb%' is 25. Between Monday and Tuesday Today is Wednesday and we come to work and find that our new data entry clerk in training has entered all new orders incorrectly on Monday and Tuesday. We want to teach our new intern how to find and correct all the erroneous records. What's the easiest way to get all the entries from the Order table entered on Monday and Tuesday? Between the position makes the task wind: SELECT ID from orders, where date BETWEEN '01/12/2018' and '01/13/2018' 26. Finding the intersection of the two tables is undoubtedly the whole reason that a relational database exists in the first place to find the relevant entries in two tables! The JOIN statement fulfills this basic S'L goal and simplifies the task. Here we are going to get a list of all entries that have matches in customer and order tables: SELECT ID from customers of INNER JOIN Orders for Customers.ID and Orders.ID The meaning of INNER JOIN, in this case, is to select entries in the customer table that have the appropriate customer ID values in the order table and return only those records. Of course, there are many types of JOIN, such as FULL, SELF and LEFT, but for now let's keep things interesting and move on to more diverse types of S'L queries. 27. Doubling power with UNION We can combine the results of two examples of S'L queries into one natural with the word UNION. Suppose we want to create a new table by combining Customer_name and phone from customers with that customer's list of recent orders so that we can look for patterns and possibly offer future purchases. Here's a quick way to accomplish the task: SELECT phone from union SELECT customers item from UNION keyword orders lets you combine JOINS and other criteria to achieve the very powerful potential of the next generation of tables. 28. Creating column labels more friendly Aliasing label speakers give us the convenience of renaming label on something more readable. There is a trade-off when naming columns to make them concise leads to a decrease in readability in subsequent daily use. In the order table, the item column contains a description of purchased products. Let's see how the item column alias is temporarily renamed for greater convenience for users: SELECT Item AS item_description from Orders 29. Always and everywhere! Wouldn't it be great if there was a set of conditions on which you could depend each time? ANY and ALL queries can make this ideal a reality! Let's see how all keywords are used to enable records only when the set of conditions is true for all entries. In the following example, we will return entries from the Order table where the idea is to get a list of large volume orders for a given product, in this case for customers who have ordered more than 50 of the product: SELECT Item from Orders where ID NO ALL (SELECT ID FROM Orders WHERE number of zgt; 50) 30. Writing friendly S'L developers is often overlooked, but a very important element of S'L scripts is adding comments to script queries to explain what it does for future developers who may need to review and update your S'L queries. The S'L script is a set of elements and commands accumulated as a file in S'L scripts. This script file can include many commands or PL/S'L codes. You can use S'L scripts to create, edit, design, execute, and delete files. - one line and /. . Multi-stage delimers allow us to add useful comments to scenarios, but this is also used in other valuable ways. Sometimes a section of code may not be used, but we don't want to delete it because we expect it to be used again. Here we can simply add a delimiter comment to disable it for a moment: / This request below is commented so that it will not perform / / SELECT item from orders where the date IS (SELECT Order_ID from orders where the number of orders is 50) / /- The request of the SDR below will be fulfilled by ignoring the text after -- - SELECT item - one comment from the orders - another comment from the orders - another comment from the I.D. ALL (SELECT ID of orders). So far, we've been researching examples of query requests for requests and combining entries from multiple queries. Now it's time to step up and look at the database at a structural level. Let's start with O'8's simplest statement, which creates a new database. Here we will create DB as a container for our customers and the order tables used in the previous ten examples above: CREATE DATABASE AllSales 32. By adding tables to our new DB Next, we'll add a table of customers that we've used in previous and then add some column labels that we're already familiar with: CREATE TABLE Customers (ID varchar (80), Name Name Phone Varchar (20), ...); While most databases are created using a user interface such as Access or OpenOffice, it's important to know how to create and delete databases and software tables using code using S'L operators. This is especially true when installing a new web application, and the user interface asks new users to enter names for DB that will be added during installation. To change or change the table value, you use changing and deleting tables using S'L ALTER Statement. In the case of relational tables with columns, the ALTER statement is used to update the table to new or amended rules or definitions. Alter belongs to the DDL team category. The data definition language can be described as a template for the commands that represent the data structures. Imagine that you decide to send a birthday card to your customers to show your appreciation for their business and so you want to add a birthday box to the customers table. In these SDS examples, you can see how easy it is to modify existing tables with alter: ALTER TABLE Customers ADD Birthday varchar (80) If the table is damaged by bad data, you can quickly delete it: DROP TABLE table_name 34. The key to successful indexing is a schematic element that includes a record for each content that enters the indexed column of a database table or cluster and gives a high-speed path to strings. There are many types of indices, such as Bitmap indices, divided indices, function-based indexes, and domain indices. Accurate indexing requires that the Core Key column contains only unique values for that purpose. This ensures that JOIN operators remain in integrity and make actual matches. Let's create our customer table again and install the ID column as the main key: CREATE TABLE Customers (ID int NOT NULL, Name varchar (80) NOT NULL AUTO_INCREMENT, PRIMARY KEY (ID); We can expand the functionality of AUTO_INCREMENT the main key so that it is automatically rescreed from the base. When it's practical, it's always best to write a list of column names in a SELECT statement rather than use a delimiter as a wildcard to select all columns. The S'L server must search and replace the operation to find all the columns in the table and write them down in a statement for you (every time SELECT is executed). For example: SELECT and from customers actually perform much faster in our database, like: SELECT Name, Birthday, Phone, Address, Zip from customer performance traps can be avoided in many ways. Avoid the time vortex of forcing the S'L server to check the system/core database each time using only the saved procedure name, and never attach it SP_. Also, the NOCOUNT ON installation reduces время, необходимое для подсчета строк, затронутых INSERT, DELETE и другими командами. Использование INNER JOIN с условием гораздо быстрее, чем использование положений WHERE с условиями. Для этого мы советуем разработчикам изучить запросы на сервере S'L на продвинутом уровне. Для производственных целей эти советы могут иметь решающее значение для адекватной производительности. Обратите внимание, что наши примеры учебника, как правило, в пользу INNER JOIN. 36. Условные результаты субкерии Оператор S'L EXISTS проверяет на наличие записей в подкерии и возвращает значение TRUE, если субкери возвращает одну или несколько записей. Взгляните на этот запрос с подповерхливым условием: SELECT Имя от клиентов, где EXISTS (SELECT Пункт от заказов, где Customers.ID и Orders.ID и цена < 50)= in= this= example= above,= the= select= returns= a= value= of= true= when= a= customer= has= orders= valued= at= less= than= $50.= 37.= copying= selections= from= table= to= table= there= are= a= hundred= and= one= uses= for= this= sql= tool.= suppose= you= want= to= archive= your= yearly= orders= table= into= a= larger= archive= table.= this= next= example= shows= how= to= do= it.= insert= into= yearly_orders= select= *= from= orders= where=> <=1 018= this= example= will= add= any= records= from= the= year= 2018= to= the= archive.= 38. catching= null= results= the= null= is= the= terminology= applied= to= describe= an= absent= value.= null= does= not= mean= zero.= a= null= value= in= a= column= of= a= table= is= a= condition= in= a= domain= that= seems= to= be= empty.= a= column= with= a= null= value= is= a= domain= with= absent= value.= it= is= essential= to= recognize= that= a= null= value= is= distinct= from= a= zero.= in= cases= where= null= values= are= allowed= in= a= field ,= calculations= on= those= values= will= produce= null= results= as= well.= this= can= be= avoided= by= the= use= of= the= ifnull= operator.= in= this= next= example,= a= value= of= zero= is= returned= rather= than= a= value= of= null= when= the= calculation= encounters= a= field= with= a= null= value:= select= item,= price= *= (qtyinstock= += ifnull(qtyonorder,= 0))= from= orders= 39. having= can= be= relieving!= the= problem= was= that= the= sql= where= clause= could= not= operate= on= aggregate= functions.= the= problem= was= solved= by= using= the= having= clause.= as= an= example,= this= next= query= fetches= a= list= of= customers= by= the= region= where= there= is= at= least= one= customer= per= region:= select= count(id),= region= from= customers= group= by= region= having= count(id)=>0; 40. Свяжите вещи с струнными! Давайте посмотрим на обработку содержимого полевых данных с помощью функций. Подстройки, вероятно, является наиболее ценным из всех встроенных функций. Это дает вам некоторую силу Regex, но это не так сложно, как Regex. Предположим, что вы хотите найти подразряд from the dots in the web address. Here's how to do it with S'L Select: SELECT SUBSTRING_INDEX (www.bytescout.com, ., 2); This line will return everything to the left of the second occurrence. , and so, in this case, he'll be back at . www.bytescout.com video to find out about each S'L request: .. and 20 more useful examples of S'L queries!! 41. Use COALESCE to return the first non-zero expression S'L Coalesce used to manage the NULL database values. In this method, NULL values are replaced by a user-defined value. The S'L Coalesce feature evaluates the series parameters and always provides the first non-zero value from the given reasoning record. SYNTAX SELECT COALESCE (NULL, NULL,'ByteScout', NULL, 'Byte') Exit ByteScout 42. Use Convert to convert any value into a specific type of data that is used to convert value into a particular type of data. For example, if you want to convert a certain value into an int datatype, you can use the conversion function to achieve that goal. For example, Syntax SELECT CONVERT (int, 27.64) Exit 27 43. DENSE_RANK ()Analytic query is an analytical query that calculates the rank of a string in an arranged collection of strings. The exit rank is the number starting from 1. DENSE_RANK is one of S'L's most important analytical queries. It returns a number of preferences as consecutive numbers. He doesn't jump rank in case of a relationship. For example, the next query will give consistent employee ranks. SELECT eno, dno, salary, DENSE_RANK () OVER (PARTITION BY Dno ORDER BY salary) AS ranking from employee; ENO DNO SALARY RATING ---------- ---------- ---------- ---------- 7933 10 1500 1 7788 10 2650 2 7831 10 6000 3 7362 20 900 1 7870 20 1200 2 7564 20 2575 3 7784 20 4000 4 7903 20 4000 4 7901 30 550 1 7655 30 1450 2 7522 30 1450 2 7844 30 1700 3 7493 30 1500 4 7698 30 2850 5 44. Query_partition_clause query_partition_clause breaks the output set into distributions or collections. The development of an analytical query is limited to the limits required by these sections related to the process, the group BY position changes the performance of the aggregate function. If query_partition_clause eliminated, the entire output collection is interpreted as a separate section. The next query applies to the OVER clause, so the average is displayed based on all outputset records. SELECT eno, dno, salary, AVG (salary) OVER () AS avg_sal OT employee; EO DNO SALARY AVG_SAL ---------- ---------- ---------- ---------- 7364 20 900 2173.21428 7494 30 1700 2173.21428 7522 30 1350 2173.21428 7567 20 20 203075 2173.21428 7652 30 1350 2173.21428 7699 30 2950 2173.21428 7783 10 2550 2173.21428 7789 20 3100 2173.21428 7838 10 5100 2173.21428 7845 30 1600 2173.21428 7877 20 1200 2173.21428 7901 30 1050 2173.21428 7903 20 3100 2173.21428 7935 10 1400 2173.21428 45. Searching for the last five entries from the table Now, if you want to get the last eight entries from the table, then it is always difficult to get such data if your table contains huge information. For example, you want the last 8 entries fr om employee table, then you can use rownum and and Offer. Rownum is temporary in S'L. For example, select a one from Employee A, where rownum 'lt;'8 union to choose from (Choose from an employee order for rowid desc), where rownum is zlt;'8; The aforementioned S'L query will give you the last eight entries from the staff table where rownum is a pseudo-column. It indexes data in the output set. LAG LAG is used to collect data from the previous series. It's an analytical function. For example, the following query gives a salary from the previous series to calculate the difference between the current line's salary and the previous line's salary. This query applies order BY to LAG. Default 1 if you don't define bias. An arbitrary default condition is given if the offset passes the window range. The default is zero if you don't determine by default. Syntax SELECT dtno, eno, emname, job, salary, LAG (sal, 1, 0) OVER (PARTITION BY dtno ORDER BY salary) AS salary_prev OT employee; Exit DTNO ENO ENAME WORK SAL SAL_PREV ---------- ---------- ---------- --------- ---------- ---------- 10 7931 STEVE CLERK 1300 0 10 7783 JOHN MANAGER 2450 1300 10 7834 KING PRESIDENT5000 2450 20 7364 ROBIN CLERK 800 0 20 7876 BRIAN CLERK 1100 800 20 7567 SHANE MANAGER 2975 2 20 7784 SCOTT ANALYST 3000 2975 20 7908 KANE ANALYST 3000 3000 30 7900 JAMES CLERK 950 0 030 7651 CONNER SELLER 1250 950 30 7522 MATTHEW SELLER 1250 1250 30 7843 VIVIAN SELLER 1500 1250 30 7494 ALLEN SELLER 1600 1500 30 7695 GLEN MANAGER 2850 1600 47. LEAD THE LEAD is also an analytical query that is applied to collect data from rows, additionally down the output of the set. The next request gives a salary from the next row to calculate the deviation between the salary of the prevailing row and the subsequent lines. Default 1 if you don't define bias. An arbitrary default condition is given if the offset passes the window range. The default is zero if you don't determine by default. SELECT eno, empname, job, salary, LEAD (salary, 1, 0) OVER (ORDER BY salary) AS salary_next, LEAD (salary, 1, 0) OVER (ORDER BY salary) - salary as salary_diff from employee; ENO EMPNAME WORK SALARY SALARY_NEXT SALARY_DIFF ---------- ---------- --------- ---------- ---------- ---------- 7369 STEVE CLERK 800 950 150 7900 JEFF CLERK 950 1100 150 7876 ADAMS CLERK 1100 1250 150 7521 JOHN SELLER 1250 1250 0 7654 MARK SELLER 1250 1300 50 7934 TANTO CLERK 1300 1500 200 7844 MATT SELLER 1500 1600 100 7499 ALEX SELLER 1600 2450 850 7782 BOON MANAGER 2450 2850 400 7698 BLAKE MANAGER 2850 2975 125 7566 JONES MANAGER 297 5 3000 25 7788 SCOTT ANALYST 3000 3000 3000 0 7902 FORD ANALYST 3000 5000 2000 7839 KING PRESIDENT 5000 0 -5000 48. PERCENT_RANK analytical PERCENT_RANK. This request requires an ORDER BY clause. Exception the section from the OVER provision determines that the entire output set is interpreted as a separate section. Teh Teh the standardised set line is 0, and the last line of the set is 1. For example, the S'L example provides the following conclusion. SELECT prdid syntax, SUM (amount), PERCENT_RANK () OVER (ORDER BY SUM (amount) DESC) AS percent_rank FROM sales group BY PRdid ORDER BY prdid; PrDID SUM (SUMMA) PERCENT_RANK ---------- ---------- ------------ 1 22623.5 0 2 223927.08 1 49. MIN Using an OVER blank position converts MIN into an analytical function. It's also an analytical query. The whole set of results is interpreted as a single section. This gives you the minimum wage for all employees and their source data. For example, the following query shows the use of MIN in select. SELECT eno, empname, dtno, salary, min (salary) OVER (PARTITION BY dtno) AS min_result OT employee; ENO EMPNAME DTNO SALARY MIN_RESULT ---------- ---------- ---------- ---------- ---------- 7782 CLARK 10 2450 1300 7839 KING 10 5000 1300 7934 MILLER 10 1300 1300 130 0 7566 JONES 20 2975 800 7902 FORD 20 3000 800 7876 ADAMS 20 1100 800 7369 SMITH 20 80 80 0 800 7788 SCOTT 20 3000 800 7521 WARD 30 1250 950 7844 TURNER 30 1500 950 7499 ALLEN 30 1600 950 7900 JAMES 30 950 950 7698 BLAKE 30 2850 950 7654 MARTIN 30 1250 950 50. MAX Using an empty over-string position converts MAX into an analytical function. The absence of a separation clause indicates that the entire output set is interpreted as a separate section. This gives maximum pay to all employees and their raw data. For example, the following query shows the use of MAX in a favorite query. SELECT eno, empname, dtno, salary, MAX (salary) OVER () AS max_result OT employee; ENO EMPNAME DTNO SALARY MAX_RESULT ---------- ---------- ---------- ---------- ---------- 7369 SMITH 20 800 3000 7499 ALLEN 30 1600 3000 7521 WARD 30 1250 3000 756 JONES 20 2975 3000 7654 MARTIN 30 1250 3000 7698 BLAKE 30 2850 3000 7782 CLARK 10 2450 30 00 7788 SCOTT 20 3000 3000 7839 KING 10 5000 3000 7844 TURNER 30 1500 3000 7876 ADAMS 20 20 20 3000 7900 JAMES 30 950 3000 7902 FORD 20 3000 3000 7934 MILLER 10 1300 3000 51. Top-N Top-N requests give the process to limit the number of lines delivered from organized data collection. They are extremely useful when users want to give the top or bottom number of rows from the table. For example, the following query gives 20 lines with 10 different values: SELECT price from sales_order ORDER BY; PRICE ---------- 100 100 200 200 300 300 400 400 500 500 600 PRICE ---------- 600 700 700 800 900 900 900 1000 1000 1000 200 20 rows selected. The CORR Analytical Function Analysis Feature is used to determine the correlation factor. This query is also used to calculate Pearson's correlation ratio. The feature calculates the following lines in a table with no zero value. This query always returns values and -1, which describe the following: Syntax: CORR (exp1, exp2) - OVER (analytic_clause) - Example SELECT empid, name, dno, salary, job, CORR (SYSDATE - joining, salary) OVER () AS my_corr_val from employee; 53. The NTILE NTILE analytical query allows users to divide the sequence set by a detailed number relative to similar groups, or containers, of the lines that authorize. If the number of rows in the collection is less than a certain number of containers, the number of containers will be reduced. The basic syntax is shown below: NTILE (exp) OVER (No partition_clause and order_by) Sample SELECT empid, name, dno, salary, NTILE (6) OVER (ORDER BY salary) AS container_no from employee; Varians, VAR_POP and VAR_SAMP the VARIANCE request, VAR_POP and VAR_SAMP are aggregated functions. They are used to determine variance, group variance and selective deviations from data collection separately. As aggregate requests or functions, they reduce the number of lines, so the expression is aggregated. If you don't have the data, we'll change the common lines in the employee table to a separate line with aggregated values. For example, the following query displays the use of these features: SELECT VARIANCE (salary) AS var_salary, as VAR_POP (salary) AS pop_salary, VAR_SAMP (salary) AS samp_salary FROM; VAR_SALARY POP_SALARY SAMP_SALARY ------------ ---------- ------------ 1479414.97 1588574.81 1388717.27 55. STDDEV, STDDEV_POP and STDDEV_SAMP STDDEV Queries, STDDEV_POP and STDDEV_SAMP cumulative requests or functions are used to determine the standard deviation, deviation of the population standard, and the cumulative standard sampling deviation individually. As aggregate queries, they reduce the number of lines, so the expression is aggregated. If the data is not located, we will convert all lines in the EMPLOYEE table into a separate line. For example, the following query shows the use of all of these features. SELECT STDDEV (salary) AS stddev_salary, STDDEV_POP (salary) AS pop_salary, STDDEV_SAMP (salary) AS samp_salary OT employee; STDDEV_SALARY POP_SALARY SAMP_SALARY ---------- ------------ ---------- 1193.50 1159.588 1193.603 If there is more than one account after the zeros fall, the STDDEV function gives the result of STDDEV_SAMP. Using an OVER blank position converts the result of a STDDEV request into an analytical query. The absence of a section indicates that the entire output set is interpreted as a specific section, so we accept the standard deviation of salary and primary data. A template that corresponds to a template corresponding to syntax adds different alternatives. The data should be treated accurately and in the proper form. THE conditions of PARTITION BY and ORDER BY of all S'L analytical queries are applied to separate data into accumulations within and within each group. If there are no sections the whole sequence set is considered to be one huge section. For example, the MEASURES position determines the result of the column that will be provided for each match. SYNtax MEASURES STRT.tstamp AS initial_tstamp, LAST (UP.tstamp) AS first_tstamp, LAST (DOWN.tstamp) AS FINISHED_TSTAMP EXAMPLE OF THE UP.PRODUCTS_SOLD AS prev (UP.products_sold), FLAT AS FLAT.products_sold - PREV (FLAT.products_sold), AS DOWN DOWN.PRODUCTS_SOLD and PREV (DOWN.products_sold) 57. FIRST_VALUE The easiest way to get analytical features is to start by studying aggregated functions. The agragion function collects or collects data from multiple lines in a unique line of results. For example, users can use AVG to earn an average of all wages in the EMPLOYEE table. Let's see how you First_Value be used. The main explanation for this FIRST_VALUE feature is shown below. Syntax: FIRST_VALUE (expr) (NULLS) (expred NULLS) OVER (analytical position) Example SELECT eno, dno, salary, FIRST_VALUE (salary) IGNORE NULLS OVER (PARTITION BY dno BY ORDER salary) AS lowest_salary_in_dept from employee; The aforementioned query will ignore zero values. LAST_VALUE the main explanation for the LAST_VALUE request or function is below. Syntax: LAST_VALUE (expr) (expr - NULLS) OVER (analytical position) LAST_VALUE analytical query is associated with the analytical function of LAST. The feature allows users to get the last exit from an organized column. Applying windows to the default output can be awesome. For example, SELECT eno, dno, salary, LAST_VALUE (salary) IGNORE NULLS OVER (PARTITION BY dno ORDER BY salary) as highest_salary_in_dept from employee; 59. The design model forecast predicts the gender and age of customers who are expected to accept the agreement card (target No. 1). THE PREDICTION function takes a price matrix correlated with design and applies to marital status, and the size of the house - to predictors. THE PREDICTION syntax can also apply some of groupING's arbitrary information to a separate model. SELECT CLIENT_GENDER, COUNT (I) AS ct, ROUND (AVG(age)) AS AVERAGE_AGE FROM MINING_DATA_SHOP WHERE PREDICTION (COST MODEL USING CLIENT_MARITAL_STATUS, house_size) - 1 GROUP BY CLIENT_GENDER ORDER BY CLIENT_GENDER; CUST_GENDER CNT AVG_AGE ------------ ---------- ---------- F 270 40 M 585 41 60. CLUSTER_SET CLUSTER_SET can get data in one of several stages: it can use a mining-type object for information, or it can extract data by performing an analytical point that creates and uses one or more moving mining patterns. This example lists the properties that have the greatest impact on cluster distribution for a 1000 customer ID. Request функции CLUSTER_DETAILS CLUSTER_SET, которые используют модель кластеризации my_sample. Пример SELECT S.cluster_id, S.cluster_id, CLUSTER_DETAILS (my_sample, S.cluster_id, 7 USING T.) kset FROM (SELECT v., CLUSTER_SET (my_sample, USING) nset FROM mining_data WHERE client_id and 1000) T, TABLE (T.nset) - ORDER BY 2 DESC; A cluster is a group table that distributes the relevant data blocks, i.e. all the tables are actually put together. For example, THE EMPLOYEE and DEPARTMENT tables are connected to the DNO column. When they are clustered, it will store all the lines in the same data blocks. About Author ByteScout The ByteScout Writers Team has a team of professional writers who own a variety of technical topics. We select the best writers to cover interesting and trendy topics for our readers. We love developers and hope that our articles will help you learn about programming and programmers. TRY PDF.CO online apps and WEB API TweetShareWhatsAppLinkedIn TweetShareWhatsAppPinLinkedIn advanced sql queries examples with answers pdf