

Version: 03.01

Date: 25-Mar-2016

Developed by:

Verified by:

Endava SQL Discipline

SQL Test

# Revision History

|  |  |  |
| --- | --- | --- |
| **Revision** | **Date of revision** | **Description of modifications** |
| 03.01 | 25-March-2016 | The third version of the document. |

# Description

**About the test:**

|  |  |
| --- | --- |
|  |  |
| Applied Level | Intermediate, Advanced |
| Number of tasks | 22 |
| Domain | Standard DB |
| Test type | PC |
| Test duration | 4 h |

**Evaluation info:**

|  |  |
| --- | --- |
|  |  |
| Evaluated person |  |
| Evaluator name |  |
| Date of evaluation |  |
| Evaluation result  (passed/failed) |  |

# Test Tasks

## Precondition:

Use the below credentials:

|  |  |
| --- | --- |
| **Server Name** | MDCH-AMWCI-S01 |
|
| **DB** | AMInternship |
|
|

## Task 1

**What are the aggregate functions? List them.**

**Answer:**

|  |
| --- |
| Aggregate functions perform a calculation on a set of values and return a single value. Except for COUNT, aggregate functions ignore null values. Aggregate functions are frequently used with the GROUP BY clause of the SELECT statement.  AVG() – returns the average value;  COUNT() – returns the number of rows;  FIRST() – returns the first value;  LAST() – returns the last value;  MAX() – returns the largest value;  MIN() – returns the smallest value;  SUM() – returns the sum. |

## Task 2

**What are the scalar functions? List them.**

**Answer:**

|  |
| --- |
| UCASE() - converts a field to upper case;  LCASE() – coverts a field to lower case;  MID() – extract characters from a text field;  LEN() – returns the length of a text field;  ROUND() - rounds a numeric field to the number of decimals specified;  NOW() – returns the current system date and time;  FORMAT() – formats how a field is to be displayed. |

## Task 3

**What is the difference between WHERE and HAVING?**

**Answer:**

|  |
| --- |
| HAVING is used to check conditions after the aggregation takes place.  WHERE is used before the aggregation takes place.  HAVING specifies a search condition for a group or an aggregate function used in SELECT statement. |

## Task 4

**Which are the differences between Function and Stored Procedure?**

**Answer:**

|  |
| --- |
| Stored Procedure (SP) | Function (UDF - User Defined |  | | Function) |  +---------------------------------+----------------------------------------+  | SP can return zero , single or | Function must return a single value |  | multiple values. | (which may be a scalar or a table). |  +---------------------------------+----------------------------------------+  | We can use transaction in SP. | We can't use transaction in UDF. |  +---------------------------------+----------------------------------------+  | SP can have input/output | Only input parameter. |  | parameter. | |  +---------------------------------+----------------------------------------+  | We can call function from SP. | We can't call SP from function. |  +---------------------------------+----------------------------------------+  | We can't use SP in SELECT/ | We can use UDF in SELECT/ WHERE/ |  | WHERE/ HAVING statement. | HAVING statement. |  +---------------------------------+----------------------------------------+  | We can use exception handling | We can't use Try-Catch block in UDF. |  | using Try-Catch block in SP. | |

## Task 5

**How many people live in India? The result should be in digits.**

**Query:**

|  |
| --- |
| select count(p.id) as 'Indian People'  from address a  inner join person p On p.address\_id=a.id  inner join city c ON c.id=a.city\_id  inner join country cr On cr.id=c.country\_id  where cr.name='India'  7 |

## Task 6

**Which is the country with the maximum number of people?**

**Query:**

|  |
| --- |
| select top 1 count(p.id), ctr.name  from  person p  inner join address a On p.address\_id=a.id  inner join city ON city.id=a.city\_id  inner join country ctr On ctr.id=city.country\_id  group by (ctr.name)  order by COUNT(p.id) DESC  Panama 11 people |

## Task 7

**Who has the biggest salary?**

**Query:**

|  |
| --- |
| select top 1 s.salary, p.first\_name,p.last\_name  from  person p  inner join job j On p.job\_id=j.id  inner join salary s ON s.id=j.salary\_id  order by (s.salary) DESC  Kevin Arnold 2996.52 |

## Task 8

**Who has the smallest salary?**

**Query:**

|  |
| --- |
| select top 1 s.salary, p.first\_name,p.last\_name  from  person p  inner join job j On p.job\_id=j.id  inner join salary s ON s.id=j.salary\_id  order by (s.salary) ASC  Keith Adams 801.05 |

## Task 9

**What is the salary average in Ankara?**

**Query:**

|  |
| --- |
| select AVG( s.salary) as 'AVG SALARY'  from  person p  inner join address a ON a.id=p.address\_id  inner join job j On p.job\_id=j.id  inner join salary s ON s.id=j.salary\_id  inner join city c ON c.id=a.city\_id  where c.name='Ankara'  2420.0114 |

## Task 10

**What is the salary average in Ankara, Panama City and San Juan?**

**Query:**

|  |
| --- |
| select AVG( s.salary) as 'AVG SALARY'  from  person p  inner join address a ON a.id=p.address\_id  inner join job j On p.job\_id=j.id  inner join salary s ON s.id=j.salary\_id  inner join city c ON c.id=a.city\_id  where c.name='Ankara' OR c.name='Panama city' OR c.name='San Juan'  2159.6128 |

## Task 11

**Which city has the biggest salary average?**

**Query:**

|  |
| --- |
| select top 1 AVG(s.salary) , c.name  from  person p  inner join address a ON a.id=p.address\_id  inner join job j On p.job\_id=j.id  inner join salary s ON s.id=j.salary\_id  inner join city c ON c.id=a.city\_id  group by (c.name)  order by AVG(s.salary) desc  2795.44 Saint-Pierre, Réunion |

## Task 12

**Display the total sum of salary id from “Job” table with salary from “Salary” table for each combination of salary id and salary.**

**Query:**

|  |
| --- |
| select j.salary\_id, sum(s.salary)  from job j  inner join salary s On s.id=j.salary\_id  group by j.salary\_id  order by sum ( j.salary\_id) asc |

## Task 13

**Display which cities have an address with different numbers.**

**Query:**

|  |
| --- |
| SELECT c.name, a.street\_num FROM address a  JOIN city c ON a.city\_id = c.id  WHERE a.street\_num IN  (SELECT a.street\_num FROM address a  JOIN city c ON a.city\_id = c.id  GROUP BY a.street\_num  HAVING COUNT(\*) < 2)  ORDER BY a.street\_num ASC |

## Task 14

**Display the top 5 frequent job titles in the “Job” table.**

**Query:**

|  |
| --- |
| select top 5 jt.title, count (j.id)  from job j  inner join job\_title jt ON j.jobtitle\_id=jt.id  group by jt.title  order by count (j.id) DESC  Systems Administrator III 23  Operator 22  Paralegal 21  Environmental Specialist 21  Assistant Professor 20 |
|  |
|  |

## Task 15

**Display the last names for which the occurrence is more than once for women born in 1988.**

**Query:**

|  |
| --- |
| select p.last\_name, count (p.id)  from person p  inner join gender g ON g.id=p.gender\_id  where g.name='female' and YEAR ( date )=1988  group by last\_name  having count(p.id)>1 |

## Task 16

**Create a new table “Title” with 2 columns: id and name. Insert in the table the personal titles used with the name.**

**Add a new column name “title\_id” (FK) in the “Person” table.**

**Query:**

|  |
| --- |
| create table Title (id int PRIMARY KEY, name varchar(50))  Alter table person  add title\_id int FOREIGN KEY REFERENCES Title(id)  INSERT INTO title  SELECT id,first\_name FROM person |

## Task 17

**Create the stored procedure which will populate random the values in the new added column of the “Person” table.**

**Query:**

|  |
| --- |
| CREATE PROCEDURE Insert\_into\_person\_titile\_id AS  Declare @count int  set @count=(SELECT COUNT(id) FROM person)  WHILE (@count>0)  BEGIN  print @count  Declare @var int  set @var= @count  Update Person  set  title\_id =FLOOR(Rand(25-10)\*10)  where id = @count  set @count=@count-1  END  EXEC Insert\_into\_person\_titile\_id |

## Task 18

**Create a new table “History\_Salary” with the columns: id, change date, person id, job id, salary old, salary new and salary diff.**

**Query:**

|  |
| --- |
| create table History\_salary (  id int,  change date,  person\_id int,  job\_id int,  salary\_old int,  salary\_new int,  salary\_diff int  ) |

## Task 19

**Create a new table “Accountancy” with the columns: id, name, salary.**

**Query:**

|  |
| --- |
| create table Accountancy (  id int,  name varchar(40),  salary int  ) |

## Task 20

**Create a new table “Statistic\_Job\_Title” with the columns: id, title, title num.**

**Query:**

|  |
| --- |
| create table Statistic\_Job\_Title (  id int,  title varchar(40),  title\_num int  ) |

## Task 21

**Create a stored procedure which will populate the “Statistic\_Job\_Title” table with the job titles and the occurrence of them from the “Job” table and will print the result set.**

**Query:**

|  |
| --- |
| CREATE PROCEDURE insert\_into\_statistic AS  delete from Statistic\_Job\_Title  insert into Statistic\_Job\_Title (title,title\_num)  select jt.title, count(jt.title)  from  job\_title jt  group by jt.title  select \* from Statistic\_Job\_Title  exec insert\_into\_statistic  Administrative Assistant IV 1  Business Systems Development Analyst 1  Computer Systems Analyst II 1  engeneer 1  Environmental Specialist 1  Environmental Tech 1  Financial Advisor 1  Geological Engineer 1  GIS Technical Architect 1  Librarian 1  manager 1  Nuclear Power Engineer 1  Occupational Therapist 1  Paralegal 1  Pharmacist 1  Physical Therapy Assistant 1  Programmer I 1  Senior Developer 1  Structural Analysis Engineer 1  Systems Administrator III 1  Tax Accountant 1  VP Accounting 1 |

## Task 22

**Create a temporary table “Number\_game” with 2 columns: Digit\_number and Letter\_number. Insert the values in the table from 1 (one) to 20 (twenty).**

**Using the table calculate the combinations of different numbers which in sum give 42.**

**Query:**

|  |
| --- |
| CREATE TABLE #Number\_game (Digit\_number INT , Letter\_number varchar(40));  insert into #Number\_game values (1,'one')  insert into #Number\_game values (2,'two')  insert into #Number\_game values (3,'three')  insert into #Number\_game values (4,'four')  insert into #Number\_game values (5,'five')  insert into #Number\_game values (6,'six')  insert into #Number\_game values (7,'seven')  insert into #Number\_game values (8,'eight')  insert into #Number\_game values (9,'nine')  insert into #Number\_game values (10,'ten')  insert into #Number\_game values (11,'eleven')  insert into #Number\_game values (12,'twelve')  insert into #Number\_game values (13,'thirteen')  insert into #Number\_game values (14,'fourteen')  insert into #Number\_game values (15,'fifteen')  insert into #Number\_game values (16,'sixteen')  insert into #Number\_game values (17,'seventeen')  insert into #Number\_game values (18,'eighteen')  insert into #Number\_game values (19,'nineteen')  insert into #Number\_game values (20,'twenty')  select  n.Digit\_number,n2.Digit\_number, n3.Digit\_number,  ( n.Digit\_number+n2.Digit\_number+n3.Digit\_number)  from #Number\_game n  CROSS JOIN #Number\_game n2  CROSS JOIN  #Number\_game n3  where ( n.Digit\_number+n2.Digit\_number+n3.Digit\_number)=42 |