|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PRUEBA TÉCNICA ASPIRANTE A ANALISTA DE DESARROLLO IVR | | | |  |
| **Nombre** | Edier Sanchez Sanchez | **Fecha** 77106018 |  | |

1. **CONOCIMIENTOS EN BASES DE DATOS (30%)**

Se tiene un modelo de bases de datos con las columnas como se muestra en la siguiente imagen:

A screenshot of a computer program

Description automatically generated with low confidence

|  |  |
| --- | --- |
| Tabla **AREAS** | |
| CODE | Código del área |
| AREA\_NAME | Nombre del área |
| BUDGET | Presupuesto del área |

|  |  |
| --- | --- |
| Tabla **EMPLOYEES** | |
| DOCUMENT | Cédula del empleado |
| FIRST\_NAME | Nombre del empleado |
| LAST\_NAME | Apellido del empleado |
| POSITION | Cargo del empleado  Ejemplo: Jefe, Líder, Analista. |
| AREA | Área del empleado  Ejemplo: IVR, IVR\_BANCO, KCRM, BPM. |
| ADMISSION\_DATE | Fecha de ingreso |

La compañía ha solicitado que se le entregue una información con base en los datos de las tablas y además realizar algunos cambios. Construya cada query para que retorne la información requerida y se realicen las modificaciones.

1. Una lista con todos los empleados, en una sola columna llamada EQUIPOS que contenga la siguiente estructura: nombre apellido (cargo – área) y esté ordenado alfabéticamente por área. **Ejemplo:** Pedro Pérez (Analista – IVR):

**SELECT CONCAT(FIRST\_NAME, ' ',LAST\_NAME, ' (' ,POSITION,' - ',AREA, ')') AS EQUIPOS**

**FROM EMPLOYEES**

**ORDER BY AREA;**

1. Todos los cargos que existen, sin repeticiones:

**SELECT DISTINCT(POSITION)**

**FROM EMPLOYEES;**

1. La cantidad de personas que hay en el área de IVR cuyo cargo es Analista:

**SELECT COUNT(\*) AS "CANTIDAD ANALISTAS IVR"**

**FROM EMPLOYEES**

**WHERE AREA = 'IVR' AND POSITION = 'ANALISTA';**

1. Obtener el presupuesto total de todas las áreas:

**SELECT SUM(BUDGET) AS "PRESUPUESTO TOTAL"**

**FROM AREAS;**

1. La cantidad de empleados que hay en cada área, ordenada de manera descendente:

**SELECT AREA, COUNT(\*) AS "CANTIDAD EMPLEADOS"**

**FROM EMPLOYEES**

**GROUP BY AREA**

**ORDER BY COUNT(\*) DESC;**

1. Obtener los documentos y nombres completos de los empleados que trabajen en áreas cuyo presupuesto se encuentre entre 46000 y 70000 USD:

**SELECT DOCUMENT AS DOCUMENTO, CONCAT(FIRST\_NAME,' ',LAST\_NAME) AS NOMBRE**

**FROM EMPLOYEES**

**INNER JOIN AREAS ON EMPLOYEES.AREA = AREAS.AREA\_NAME**

**WHERE AREA IN (SELECT AREA\_NAME**

**FROM AREAS**

**WHERE BUDGET BETWEEN 46000 AND 70000);**

Debido a una reestructuración de la compañía se deben hacer los siguientes cambios:

1. Cambie los empleados del área IVR BANCO al área de IVR:

**UPDATE EMPLOYEES SET AREA = 'IVR'**

**WHERE AREA = 'IVR BANCO';**

1. Elimine los empleados que lleven 1 año o menos en la compañía:

**DELETE FROM EMPLOYEES**

**WHERE ADMISSION\_DATE >= DATE\_SUB(DATE(NOW()),INTERVAL 1 YEAR);**

1. **CONOCIMIENTOS DE GIT. (10%)**

Indique los comandos de git para:

* 1. Crear un nuevo repositorio local para un proyecto llamado KSF\_PRUEBA\_IVR:

**git init KSF\_PRUEBA\_IVR**

* 1. Enlazar este repositorio local con el repositorio remoto:

**git remote add origin <URL\_del\_repositorio\_remoto>**

* 1. Crear y cambiarse a una rama llamada IVR-{{CEDULA}}:

**git branch IVR-{{CEDULA}}**

**git checkout IVR-{{CEDULA}}** ó en una sóla línea de comando:

**git checkout -b IVR-{{CEDULA}}**

* 1. Realizar un commit (suponga que sí hay archivos modificados en su repositorio local):
* *Primero agrego los archivos al área de preparación (stage)*
* *y luego realizo el commit.*

**git add .**

**git commit -m "Mensaje del commit"**

* 1. Subir el contenido del último commit del repositorio local al repositorio remoto:

**git push origin <nombre\_de\_la\_rama>**

* 1. Solicitar la fusión de la rama IVR-{{CEDULA}} con la rama master:

**git merge IVR-{{CEDULA}}**

* 1. Si la rama estuviera desactualizada, indique el comando para actualizar su repositorio local:

**git pull origin <nombre\_de\_la\_rama>**

1. **CASO DE USO PROGRAMACIÓN JAVA (30%)**

Desarrolle un código que reciba una fecha en el formato yyyy-mm-dd, de tal manera que aplique la lógica necesaria para devolver la fecha de pago de la quincena, tal que esta sea el día hábil más cercano al día 15 o 30 con respecto a lo ingresado por el usuario, dando prioridad al anterior en caso de empate.

Nota: Puede hacer uso de los elementos externos que crea necesarios (base de datos, archivos, etc.)

Ej: Si el usuario ingresa 30 de abril que es domingo, y el próximo lunes es festivo, la fecha a devolver será el viernes 28.

Si el usuario ingresa el día 30 de julio que es domingo y el lunes no es festivo, devolverá el lunes 31.

Si el usuario ingresa el día 15 de junio que es jueves, devolverá el mismo día.

Formatos de entrega:

* + - 1. Repositorio en Git Hub
      2. Proyecto comprimido en .zip
      3. Archivo de texto plano.

**Rta:**

Proyecto comprimido en .zip:

***Doble click sobre el archivo abrir:***



1. **EJERCICIO DE LÓGICA DE PROGRAMACIÓN. (30%)**

Desarrolle el diagrama de flujo de un algoritmo que reciba n valores (números enteros entre el 0 y el 1000) por consola, sin importar un orden, y que los organice de manera ascendente.

Formatos de entrega: Imagen con el diagrama realizado.

**Rta:**

Diagrama de flujo desarrollado en Flowgorithm.

Se adjunta diagrama de flujo en archivo pdf e imagen:

***Doble click sobre el archivo para abrir:***



