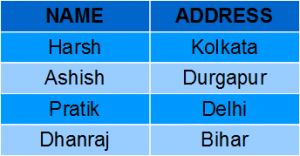
1. **Explain different types of views. Demonstrate with suitable examples.**

Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition. Example [](https://user-images.githubusercontent.com/88320437/153807269-0f988104-86df-46a5-9fc0-64bc4d749390.png) Creating View from a single table: In this example we will create a View named DetailsView from the table StudentDetails. Query: CREATE VIEW DetailsView AS SELECT NAME, ADDRESS FROM StudentDetails WHERE S\_ID < 5; To see the data in the View, we can query the view in the same manner as we query a table. SELECT \* FROM DetailsView; [](https://user-images.githubusercontent.com/88320437/153807394-e8ab6187-b23a-45d1-8f74-f70d8a03c588.png)

1. **What is the difference between function and stored procedure? Write syntax for creating functions and stored procedures.**

Difference between functions and stored procedures in PL/SQL Differences between Stored procedures(SP) and Functions(User-defined functions (UDF)):

Function 1.A function has a return type and returns a value. 2.You cannot use a function with Data Manipulation queries. Only Select queries are allowed in functions. 3.A function does not allow output parameters. 4.You cannot manage transactions inside a function. 5.You cannot call stored procedures from a function. 6.You can call a function using a select statement. Procedure 1.A procedure does not have a return type. But it returns values using the OUT parameters. 2.You can use DML queries such as insert, update, select etc… with procedures. 3.A procedure allows both input and output parameters. 4.You can manage transactions inside a procedure. 5.You can call a function from a stored procedure.

1. **What is an index in SQL? What are the different types of indexes in SQL?**

SQL Indexes are nothing but optional structure associated with the table which may or may not improve the performance of Query” 1.Normal index 2.Unique Index 3.Bit Map Index 4.Composite Index 5.B-Tree Index(Oracle considered Normal indexes as B-Tree Indexes) 6.Function Based Index 7.Clustered Index 8.Non-Clustered Index.

CREATE TABLE sales.persons ( person\_id INT PRIMARY KEY IDENTITY, first\_name NVARCHAR(100) NOT NULL, last\_name NVARCHAR(100) NOT NULL );

CREATE TABLE sales.deals ( deal\_id INT PRIMARY KEY IDENTITY, person\_id INT NOT NULL, deal\_note NVARCHAR(100), FOREIGN KEY(person\_id) REFERENCES sales.persons( person\_id) );

insert into sales.persons(first\_name, last\_name) values ('John','Doe'), ('Jane','Doe');

insert into sales.deals(person\_id, deal\_note) values (1,'Deal for John Doe');

Next, create a new stored procedure named usp\_report\_error that will be used in a CATCH block to report the detailed information of an error:

CREATE PROC usp\_report\_error AS SELECT  
ERROR\_NUMBER() AS ErrorNumber  
,ERROR\_SEVERITY() AS ErrorSeverity  
,ERROR\_STATE() AS ErrorState  
,ERROR\_LINE () AS ErrorLine  
,ERROR\_PROCEDURE() AS ErrorProcedure  
,ERROR\_MESSAGE() AS ErrorMessage;  
GO Code language: SQL (Structured Query Language) (sql) Then, develop a new stored procedure that deletes a row from the sales.persons table:

CREATE PROC usp\_delete\_person( @person\_id INT ) AS BEGIN BEGIN TRY BEGIN TRANSACTION; -- delete the person DELETE FROM sales.persons WHERE person\_id = @person\_id; -- if DELETE succeeds, commit the transaction COMMIT TRANSACTION;  
END TRY BEGIN CATCH -- report exception EXEC usp\_report\_error;

-- Test if the transaction is uncommittable.

IF (XACT\_STATE()) = -1

BEGIN

PRINT N'The transaction is in an uncommittable state.' +

'Rolling back transaction.'

ROLLBACK TRANSACTION;

END;

-- Test if the transaction is committable.

IF (XACT\_STATE()) = 1

BEGIN

PRINT N'The transaction is committable.' +

'Committing transaction.'

COMMIT TRANSACTION;

END;

END CATCH

END; GO EXEC usp\_delete\_person 2; EXEC usp\_delete\_person 1;