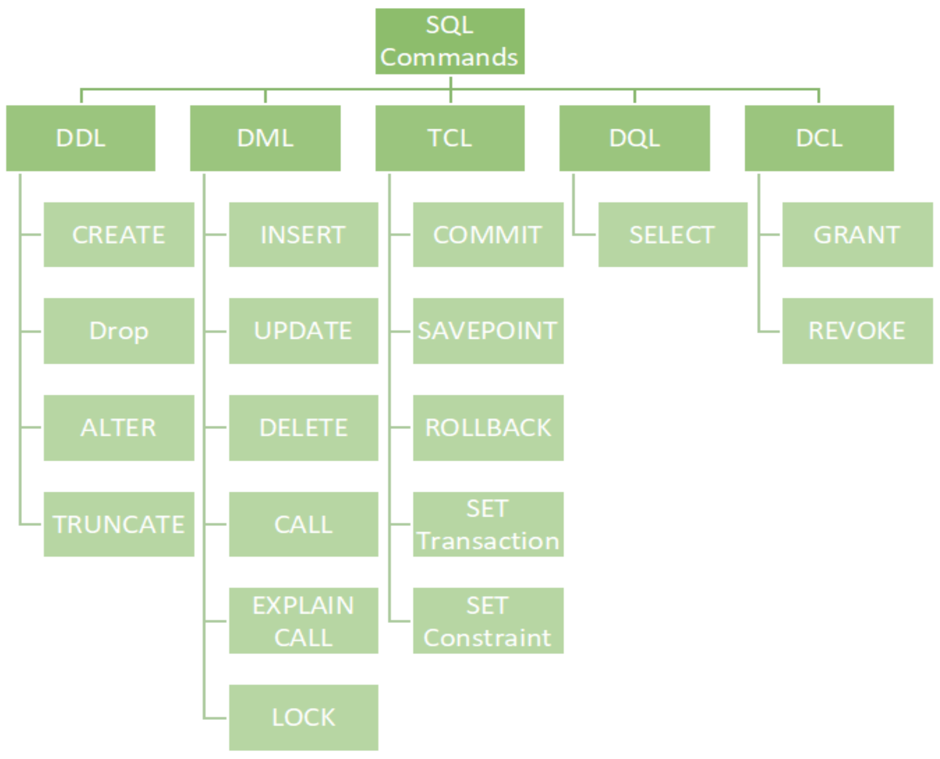
SQL OVERVIEW

SQL stands for Structure Query Language used to store, manipulate and retrieve data from the databases.

RDBMS stands for Relational Database Management System and is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
The data in RDBMS is stored in database objects called tables. A table is a collection of related data entries and it consists of columns and rows.

SQL Query Commands:



In SQL, commands are mainly categorized into four/five categories as:

1. **DDL (Data Definition Language):**
   DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database. DDL is a set of SQL commands used to create, modify, and delete database structures but not data.

List of DDL commands:

- **CREATE:** This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).
- **DROP:** This command is used to delete objects from the database.
- **ALTER:** This is used to alter the structure of the database.
- **TRUNCATE:** This is used to remove all records from a table, including all spaces allocated for the records are removed.
- **COMMENT:** This is used to add comments to the data dictionary.
- **RENAME:** This is used to rename an object existing in the database.

2. **DQL (Data Query Language):**
DQL statements are used for performing queries on the data within schema objects. The purpose of the DQL Command is to get some schema relation based on the query passed to it. It includes the SELECT statement. This command allows getting the data out of the database to perform operations with it. When a SELECT is fired against a table or tables the result is compiled into a further temporary table, which is displayed or perhaps received by the program i.e., a front-end.

List of DQL:

- **SELECT:** It is used to retrieve data from the database.

3. **DML (Data Manipulation Language):**
The SQL commands that deal with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements. It is the component of the SQL statement that controls access to data and to the database. Basically, DCL statements are grouped with DML statements.

List of DML commands:

- **INSERT:** It is used to insert data into a table.
- **UPDATE:** It is used to update existing data within a table.
- **DELETE:** It is used to delete records from a database table.
- **LOCK:** Table control concurrency.
- **CALL:** Call a PL/SQL or JAVA subprogram.
- **EXPLAIN PLAN:** It describes the access path to data.

4. **DCL (Data Control Language):**
DCL includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions, and other controls of the database system.

List of DCL commands:

- **GRANT:** This command gives users access privileges to the database.
- **REVOKE:** This command withdraws the user's access privileges given by using the GRANT command.

5. **TCL (Transaction Control Language):**
   TCL commands deal with the transaction within the database.

   List of TCL commands:
   - **COMMIT:** Commits a Transaction.
   - **ROLLBACK:** Rollbacks a transaction in case of any error occurs.
   - **SAVEPOINT:** Sets a save point within a transaction.
   - **SET TRANSACTION:** Specify characteristics for the transaction.

DATABASE CREATION, DROP and BACKUP:

Before getting started to write query or manipulate data, we have to create databases where we have to store tables and finally records into them.
After creating database, we have to select the database in which we have to work or querying out items from. A database is like a 'folder' in a system that we use to create in order to keep different files. We can delete/drop the database if we want to do so by executing DROP statement and also if we want the database to be backup, we can do that also by executing BACKUP command.

Syntax:

```
CREATE DATABASE database_name;
USE DATABASE database_name;
BACKUP DATABASE database_name TO DISK= 'file_path';
DROP DATABASE database_name;
```

CREATING TABLE:

After creating database, we have to create tables where we can maintain our data. A table is like a 'file' which we kept in folders in our system. A table consist of rows and columns where we can store data.

Syntax:

```
CREATE TABLE table_name
        ( Column_1  column_type constrain,
          Column_2  column_type constrain
          Column_3  column_type ,....
          Column_n  column_type );
```

Before creating table, we have to know about the constraints and data types:

SQL Constraints:
SQL constraints are used to specify rules for the data in a table. Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.
Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

- NOT NULL - Ensures that a column cannot have a NULL value
- UNIQUE - Ensures that all values in a column are different
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- FOREIGN KEY - Prevents actions that would destroy links between tables
- CHECK - Ensures that the values in a column satisfies a specific condition
- DEFAULT - Sets a default value for a column if no value is specified
- CREATE INDEX - Used to create and retrieve data from the database very quickly

DATA Types:
Each column in a database table is required to have a name and a data type.
An SQL developer must decide what type of data that will be stored inside each column when creating a table. The data type is a guideline for SQL to understand what type of data is expected inside of each column, and it also identifies how SQL will interact with the stored data. These are INT, CHAR, VARCHAR, DATE, DECIMAL, STRING, BOOL, YEAR, DOUBLE etc. Check official documentation for more data types and conventions.

INSERTING DATA INTO TABLES:

For insertion of data into the table, we have to use INSERT INTO command to update records into table.

Manual updating data in SQL using INSERT statement

```
INSERT INTO table_name
        (table_column_1, table_column_2,………, table_column_n)
        values
        (data_01, data_02, data_03, ………, data_0n),
        (data_11, data_12, data_13, ………, data_1n),
        (data_21, data_22, data_23, ………, data_2n),
        (data_31, data_32, data_33, ………, data_3n)
```

Loading data in SQL from external files

```
LOAD DATA INFILE
'D:/SampleDataSet.csv'        ~~~~file location in the system~~~~
INTO TABLE table_name
FIELDS TERMINATED by ','
ENCLOSED by '"'
lines terminated by '\n'
IGNORE 1 ROWS;                ~~~~for not including headers row~~~~
```

UPDATING, DROP AND ADDING TABLE STRUCTURE:

For updating data, we have to use UPDATE and ALTER command as per requirement. For deleting particular column, we can use DROP command along with the ALTER TABLE command. And for adding columns to our existing table, we have to use ADD command along with the ALTER TABLE command.

Syntax:

```
ALTER TABLE table_name MODIFY COLUMN column_name data_type ;
ALTER TABLE table_name DROP COLUMN column_name;
ALTER TABLE table_name ADD COLUMN column_name data_type ;
```

DELETE, DROP AND TRUNCATE TABLE STRUCTURE:

These commands are very useful and powerful when it comes to data maintenance.
DELETE command is use to delete the records based on certain conditions.
TRUNCATE command is use to wipe out entire data from the table leaving behind the schema of the table.
DROP command is use to wipe out data along with the schema.

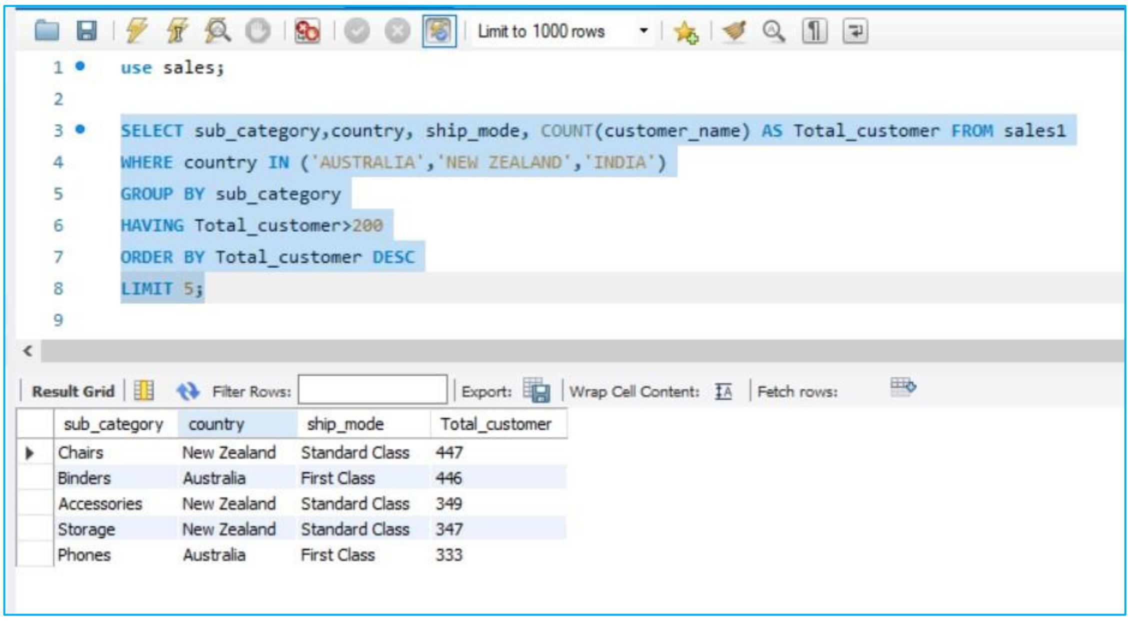
Syntax:

```
DELETE TABLE table_name where condition;
TRUNCATE TABLE table_name;
DROP TABLE table_name;
```

GENERAL QUERY, SUB-QUERY AND ORDER OF STATEMENTS:

A general query is the set of instruction that we execute to fetch the desired result. When a query is run inside a query, its known as sub-query. Some of the general query commands are as follow:
SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY, LIMIT.

Syntax:
SELECT * FROM table_name
WHERE condition
GROUP BY non_aggregate_column
HAVING aggregate_condition
ORDER BY column
LIMIT limiting_value;


```
use sales;

SELECT sub_category,country, ship_mode, COUNT(customer_name) AS Total_customer FROM sales1
WHERE country IN ('AUSTRALIA','NEW ZEALAND','INDIA')
GROUP BY sub_category
HAVING Total_customer>200
ORDER BY Total_customer DESC
LIMIT 5;
```

sub_category	country	ship_mode	Total_customer
Chairs	New Zealand	Standard Class	447
Binders	Australia	First Class	446
Accessories	New Zealand	Standard Class	349
Storage	New Zealand	Standard Class	347
Phones	Australia	First Class	333

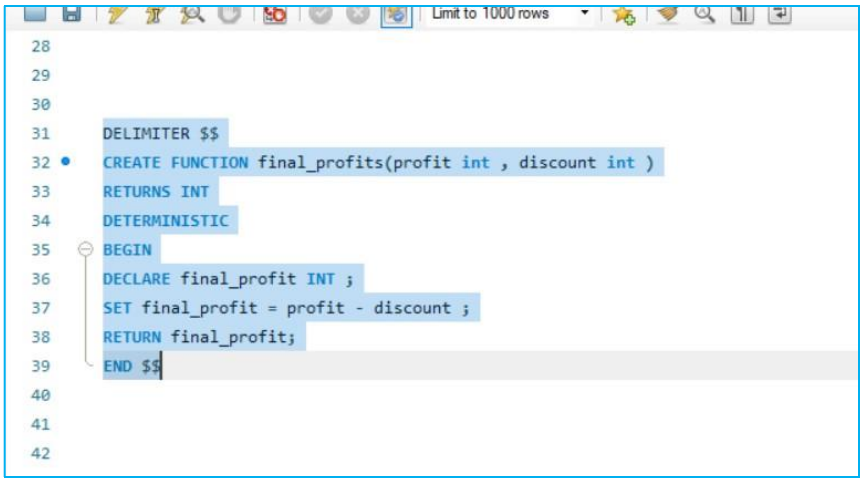
Order of execution of commands in query:

Order	Clause	Function
1	FROM	Tables are joined to get the base data.
2	WHERE	The base data is filtered.
3	GROUP BY	The filtered base data is grouped.
4	HAVING	The grouped base data is filtered.
5	SELECT	The final data is returned.
6	ORDER BY	The final data is sorted.
7	LIMIT	The returned data is limited to row count.

PROCEDURE AND FUNCTIONS:

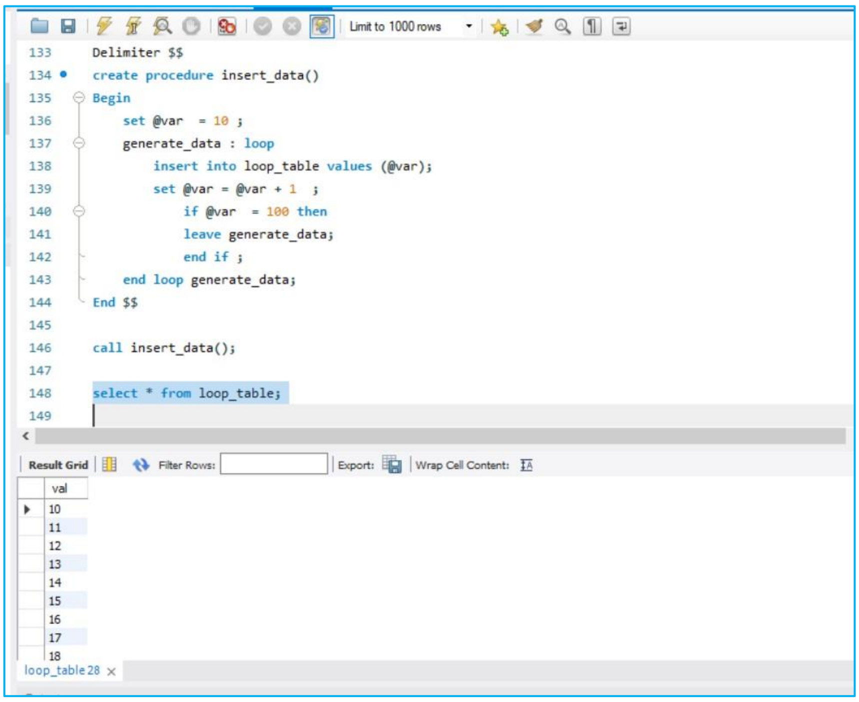
A procedures or function is a group or set of SQL statements that perform a specific task.
A function can be utilize in select statement while procedure has to be call using CALL/EXCUTE statement. The major difference between a procedure and a function is, a function must always return a value, but a procedure may or may not return a value.

This is an example where we have created a function which provide final profit by taking considering the values like profit and discount columns.



```
DELIMITER $$
CREATE FUNCTION final_profits(profit int , discount int )
RETURNS INT
DETERMINISTIC
BEGIN
DECLARE final_profit INT ;
SET final_profit = profit - discount ;
RETURN final_profit;
END $$
```

This is an example where we have created a procedure to store the data inside our "loop_table" where we have inserted a column with numbers starting from 10 to 100 and incremented by 1.



```
Delimiter $$
create procedure insert_data()
Begin
    set @var  = 10 ;
    generate_data : loop
        insert into loop_table values (@var);
        set @var = @var + 1  ;
            if @var  = 100 then
            leave generate_data;
            end if ;
    end loop generate_data;
End $$

call insert_data();

select * from loop_table;
```

val
10
11
12
13
14
15
16
17
18

loop_table 28

WINDOWS FUNCTION:

Based on application we have a wide list of windows function available in SQL. Window functions are pre-defined function store in SQL. We define Windows functions by using OVER clause.

Types of different window functions.

- Aggregate Window Functions
  SUM(), MAX(), MIN(), AVG(), COUNT()
- Ranking Window Functions
  RANK(), DENSE_RANK(), ROW_NUMBER(), NTILE()
- Value Window Functions
  LAG(), LEAD(), FIRST_VALUE(), LAST_VALUE()

TRIGGERS:

Triggers are the stored procedure which get execute after/before a certain event defined in the procedure.

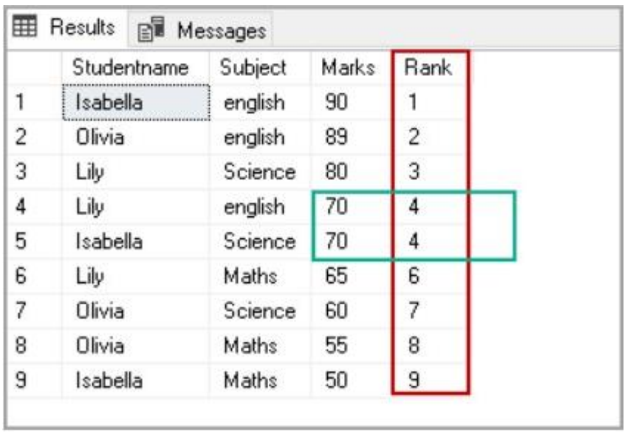
LIMIT FUNCTION, TOP Nth POSITION:

By using LIMIT clauses, we can limit our output and can get the nth position of the records order by ascending or descending values.

RANK, DENSE RANK FUNCTION:

Rank, Dense rank function are used to find the rank of the records. If values are repeated then rank function gives the same rank for the similar values and skipping the succeeding rank while dense rank gives the same rank for similar values and will not skip any rank.

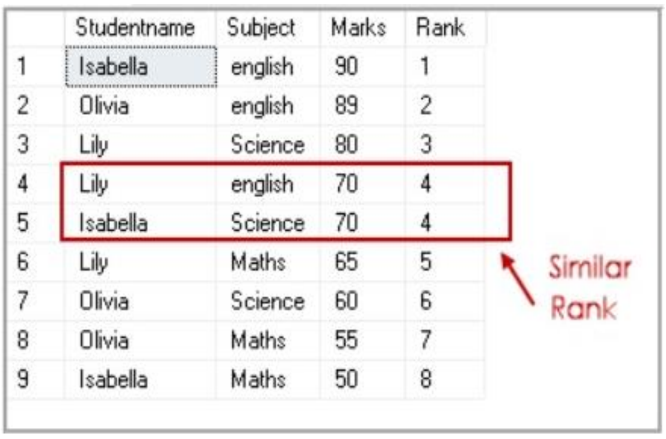
RANK()



Results | Messages

	Studentname	Subject	Marks	Rank
1	Isabella	english	90	1
2	Olivia	english	89	2
3	Lily	Science	80	3
4	Lily	english	70	4
5	Isabella	Science	70	4
6	Lily	Maths	65	6
7	Olivia	Science	60	7
8	Olivia	Maths	55	8
9	Isabella	Maths	50	9

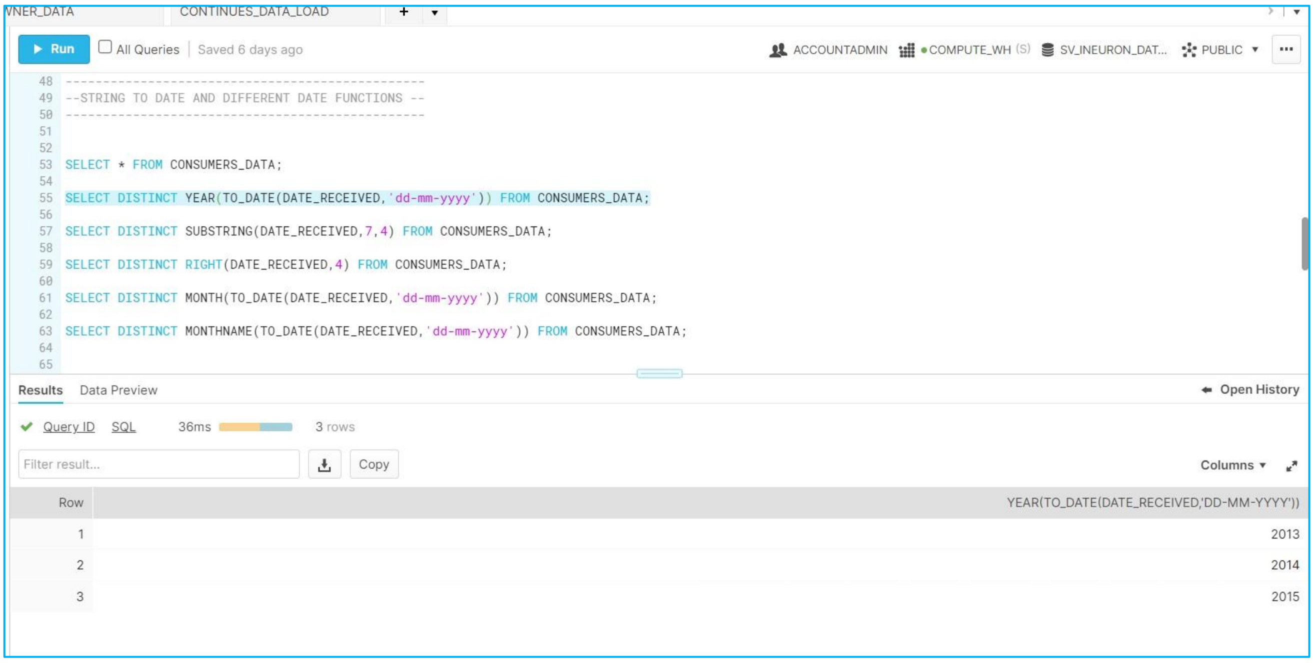
DENSE_RANK()



	Studentname	Subject	Marks	Rank
1	Isabella	english	90	1
2	Olivia	english	89	2
3	Lily	Science	80	3
4	Lily	english	70	4
5	Isabella	Science	70	4
6	Lily	Maths	65	5
7	Olivia	Science	60	6
8	Olivia	Maths	55	7
9	Isabella	Maths	50	8

DATE FUNCTION:

A SQL contains wide variety of date function and are very much used in an industry. There are various commands related to date function whether its extraction of year, month, day, quarter or to typecast the strings to date function.

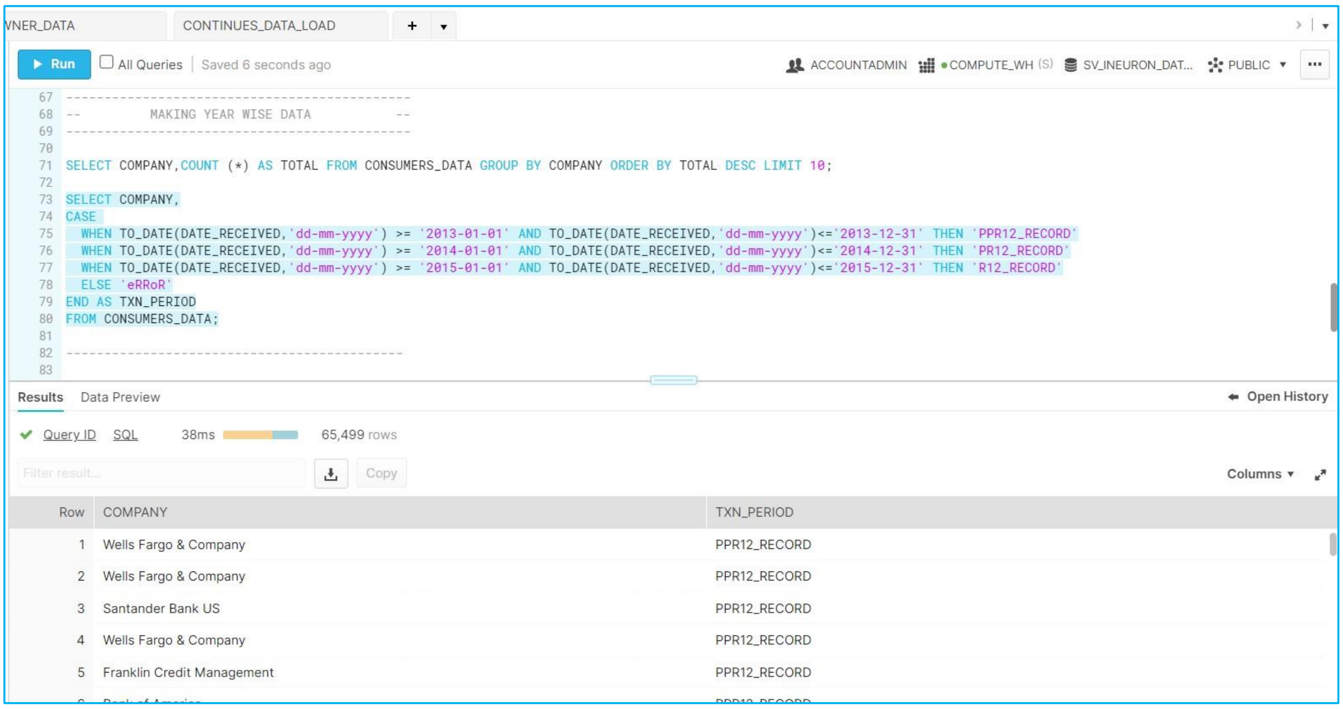


```
---------------------------------------------
--STRING TO DATE AND DIFFERENT DATE FUNCTIONS --
---------------------------------------------


SELECT * FROM CONSUMERS_DATA;

SELECT DISTINCT YEAR(TO_DATE(DATE_RECEIVED,'dd-mm-yyyy')) FROM CONSUMERS_DATA;

SELECT DISTINCT SUBSTRING(DATE_RECEIVED,7,4) FROM CONSUMERS_DATA;

SELECT DISTINCT RIGHT(DATE_RECEIVED,4) FROM CONSUMERS_DATA;

SELECT DISTINCT MONTH(TO_DATE(DATE_RECEIVED,'dd-mm-yyyy')) FROM CONSUMERS_DATA;

SELECT DISTINCT MONTHNAME(TO_DATE(DATE_RECEIVED,'dd-mm-yyyy')) FROM CONSUMERS_DATA;
```

Row	YEAR(TO_DATE(DATE_RECEIVED,'DD-MM-YYYY'))
1	2013
2	2014
3	2015

CASE STATEMENT:

The CASE statement is just like SWTICH case in programming, goes through conditions and returns a value when the first condition is met. So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause.

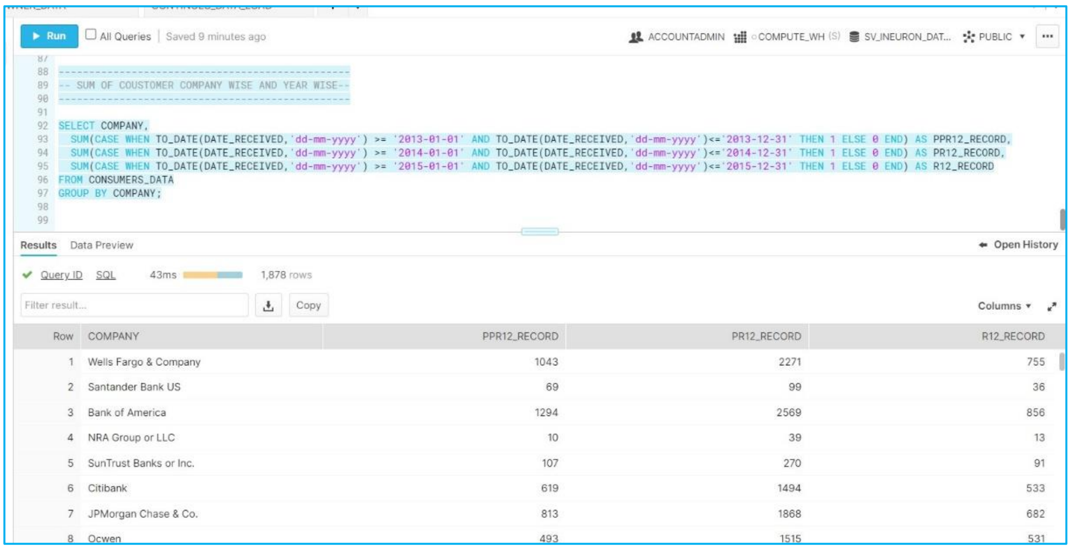


```
---------------------------------------------
--          MAKING YEAR WISE DATA          --
---------------------------------------------

SELECT COMPANY,COUNT (*) AS TOTAL FROM CONSUMERS_DATA GROUP BY COMPANY ORDER BY TOTAL DESC LIMIT 10;

SELECT COMPANY,
CASE
  WHEN TO_DATE(DATE_RECEIVED,'dd-mm-yyyy') >= '2013-01-01' AND TO_DATE(DATE_RECEIVED,'dd-mm-yyyy')<='2013-12-31' THEN 'PPR12_RECORD'
  WHEN TO_DATE(DATE_RECEIVED,'dd-mm-yyyy') >= '2014-01-01' AND TO_DATE(DATE_RECEIVED,'dd-mm-yyyy')<='2014-12-31' THEN 'PR12_RECORD'
  WHEN TO_DATE(DATE_RECEIVED,'dd-mm-yyyy') >= '2015-01-01' AND TO_DATE(DATE_RECEIVED,'dd-mm-yyyy')<='2015-12-31' THEN 'R12_RECORD'
  ELSE 'eRRoR'
END AS TXN_PERIOD
FROM CONSUMERS_DATA;

---------------------------------------------
```

Row	COMPANY	TXN_PERIOD
1	Wells Fargo & Company	PPR12_RECORD
2	Wells Fargo & Company	PPR12_RECORD
3	Santander Bank US	PPR12_RECORD
4	Wells Fargo & Company	PPR12_RECORD
5	Franklin Credit Management	PPR12_RECORD

PIVOTING:

Pivot table command is available in SQL to do the pivot and unpivot the records. But the core concept of pivoting is taken from CASE statement.



```
--------------------------------------------------
-- SUM OF COUSTOMER COMPANY WISE AND YEAR WISE--
--------------------------------------------------

SELECT COMPANY,
  SUM(CASE WHEN TO_DATE(DATE_RECEIVED,'dd-mm-yyyy') >= '2013-01-01' AND TO_DATE(DATE_RECEIVED,'dd-mm-yyyy')<='2013-12-31' THEN 1 ELSE 0 END) AS PPR12_RECORD,
  SUM(CASE WHEN TO_DATE(DATE_RECEIVED,'dd-mm-yyyy') >= '2014-01-01' AND TO_DATE(DATE_RECEIVED,'dd-mm-yyyy')<='2014-12-31' THEN 1 ELSE 0 END) AS PR12_RECORD,
  SUM(CASE WHEN TO_DATE(DATE_RECEIVED,'dd-mm-yyyy') >= '2015-01-01' AND TO_DATE(DATE_RECEIVED,'dd-mm-yyyy')<='2015-12-31' THEN 1 ELSE 0 END) AS R12_RECORD
FROM CONSUMERS_DATA
GROUP BY COMPANY;
```

Query ID SQL 43ms 1,878 rows

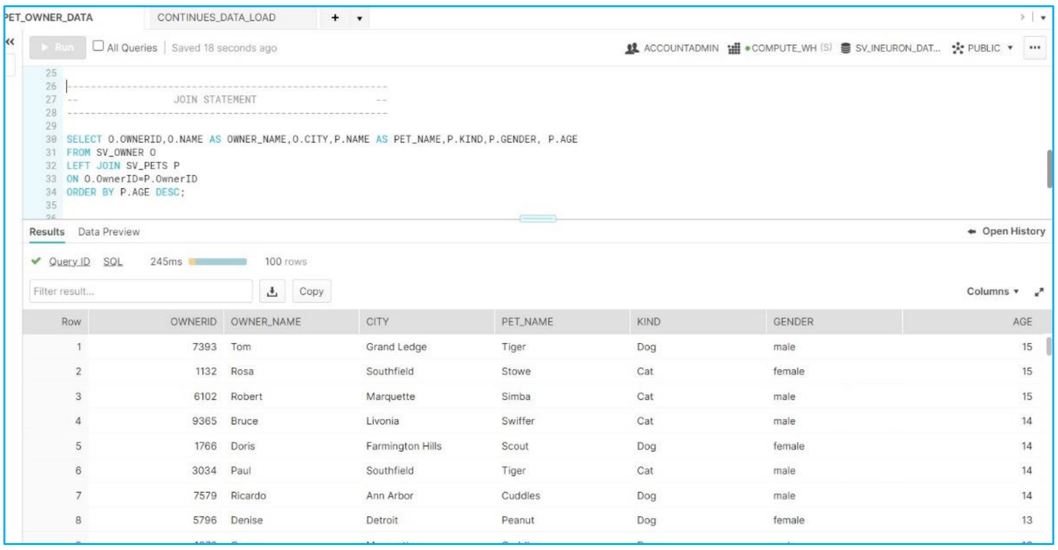
Row	COMPANY	PPR12_RECORD	PR12_RECORD	R12_RECORD
1	Wells Fargo & Company	1043	2271	755
2	Santander Bank US	69	99	36
3	Bank of America	1294	2569	856
4	NRA Group or LLC	10	39	13
5	SunTrust Banks or Inc.	107	270	91
6	Citibank	619	1494	533
7	JPMorgan Chase & Co.	813	1868	682
8	Ocwen	493	1515	531

JOINS:

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

Following are the different types of the JOINs in SQL:

- (INNER) JOIN: Returns records that have matching values in both tables
- LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
- RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
- FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table



```
------------------------------------------------------------
--                    JOIN STATEMENT                      --
------------------------------------------------------------

SELECT O.OWNERID,O.NAME AS OWNER_NAME,O.CITY,P.NAME AS PET_NAME,P.KIND,P.GENDER, P.AGE
FROM SV_OWNER O
LEFT JOIN SV_PETS P
ON O.OwnerID=P.OwnerID
ORDER BY P.AGE DESC;
```

Query ID SQL 245ms 100 rows

Row	OWNERID	OWNER_NAME	CITY	PET_NAME	KIND	GENDER	AGE
1	7393	Tom	Grand Ledge	Tiger	Dog	male	15
2	1132	Rosa	Southfield	Stowe	Cat	female	15
3	6102	Robert	Marquette	Simba	Cat	male	15
4	9365	Bruce	Livonia	Swiffer	Cat	male	14
5	1766	Doris	Farmington Hills	Scout	Dog	female	14
6	3034	Paul	Southfield	Tiger	Cat	male	14
7	7579	Ricardo	Ann Arbor	Cuddles	Dog	male	14
8	5796	Denise	Detroit	Peanut	Dog	female	13