SQL SCRIP MATILLION SESSION 11

```
show file formats;

/*

The SQL EXCEPT statement is used to filter records based on the intersection of records returned via two SELECT statements.

The records that are common between the two tables are filtered from the table on the left side of the SQL EXCEPT statement

and the remaining records are returned.

*/


CREATE OR REPLACE TABLE Books1

(

   id INT,

   name VARCHAR(50) NOT NULL,

   category VARCHAR(50) NOT NULL,

   price INT NOT NULL

 );


CREATE OR REPLACE TABLE Books2

(

   id INT,

   name VARCHAR(50) NOT NULL,

   category VARCHAR(50) NOT NULL,

   price INT NOT NULL

 );


CREATE OR REPLACE TABLE Book_Combined

(

   id INT,

   name VARCHAR(50) NOT NULL,
```

SQL SCRIP MATILLION SESSION 11

```
   category VARCHAR(50) NOT NULL,
   price INT NOT NULL
);

INSERT INTO Books1
VALUES
(1, 'Book1', 'Cat1', 1800),
(2, 'Book2', 'Cat2', 1500),
(3, 'Book3', 'Cat3', 2000),
(4, 'Book4', 'Cat4', 1300),
(5, 'Book5', 'Cat5', 1500),
(6, 'Book6', 'Cat6', 5000),
(7, 'Book7', 'Cat7', 8000),
(8, 'Book8', 'Cat8', 5000),
(9, 'Book9', 'Cat9', 5400),
(10, 'Book10', 'Cat10', 3200);

SELECT * FROM Books1;

INSERT INTO Books2
VALUES
(6, 'Book6', 'Cat6', 5000),
(7, 'Book7', 'Cat7', 8000),
(8, 'Book8', 'Cat8', 5000),
(9, 'Book9', 'Cat9', 5400),
(10, 'Book10', 'Cat10', 3200),
(11, 'Book11', 'Cat11', 5000),
(12, 'Book12', 'Cat12', 8000),
(13, 'Book13', 'Cat13', 5000),
```

SQL SCRIP MATILLION SESSION 11

(14, 'Book14', 'Cat14', 5400),

(15, 'Book15', 'Cat15', 3200);

SELECT * FROM Books2;

SELECT id, name, category, price FROM Books1

Except

SELECT id, name, category, price FROM Books2;

SELECT id, name, category, price FROM Books2

Except

SELECT id, name, category, price FROM Books1;

SELECT id, name, category, price FROM Books1

INTERSECT

SELECT id, name, category, price FROM Books2;

/* EXCEPT vs NOT NULL

Now that you know how an EXCEPT statement works, it is important to understand the difference between

SQL EXCEPT statement and NOT IN statement.

There are two major differences:

-- The EXCEPT statement only returns the distinct records, whereas a NOT IN statement returns all the records that are not filtered by the NOT IN statement

--In the EXCEPT statement, the comparison between two SELECT statements is based on all the columns in both the tables.

While a NOT IN statement compares values from a single column */

SQL SCRIP MATILLION SESSION 11

-- Here is an example of how a NOT IN statement can be used to filter all records from the Books1 table, that also exist in the Books2 table:

SELECT id, name, category, price FROM Books1

WHERE id NOT IN (SELECT id from Books2);

/* When comparing UNION vs. UNION ALL, there is one major difference:

-- UNION only returns unique

--UNION ALL returns all records, including duplicates. */

-- The WHERE clause shown is an optional inclusion:

SELECT column_1, column_2

FROM table_1

[WHERE condition]

UNION / UNION ALL

SELECT column_1, column_2

FROM table_2

[WHERE condition];

/*

There are several usage rules for UNION and UNION ALL.

Failure to adhere to these rules will result in error messages:

-- The number of columns used in your first query and your second query must be the same and the data types (e.g. INT, VARCHAR, etc.) must match.

SQL SCRIP MATILLION SESSION 11

-- The column names included in both queries can differ; when this is the case, the resulting dataset will show the column names from the first query.

-- When using SQL aliases in conjunction with UNION and UNION ALL, you only need to include the alias with the first query.

Including it with the second query will not cause an error, but it will also have no impact on the output.

You can use UNION and UNION ALL to combine tables that do not have any columns in common.

*/

```
SELECT id, name, category, price FROM BOOKS1

UNION

SELECT id, name, category, price FROM BOOKS2

ORDER BY id;


SELECT id, name, category, price FROM BOOKS1

UNION ALL

SELECT id, name, category, price FROM BOOKS2

ORDER BY id;


CREATE OR REPLACE TABLE AGENTS

  (

   "AGENT_CODE" CHAR(6) NOT NULL PRIMARY KEY,

         "AGENT_NAME" CHAR(40),

         "WORKING_AREA" CHAR(35),

         "COMMISSION" NUMBER(10,2),

         "PHONE_NO" CHAR(15),

         "COUNTRY" VARCHAR2(25)

          );


INSERT INTO AGENTS VALUES ('A007', 'Ramasundar', 'Bangalore', '0.15', '077-25814763', '');
```

SQL SCRIP MATILLION SESSION 11

```
INSERT INTO AGENTS VALUES ('A003', 'Alex ', 'London', '0.13', '075-12458969', '');
INSERT INTO AGENTS VALUES ('A008', 'Alford', 'New York', '0.12', '044-25874365', '');
INSERT INTO AGENTS VALUES ('A011', 'Ravi Kumar', 'Bangalore', '0.15', '077-45625874', '');
INSERT INTO AGENTS VALUES ('A010', 'Santakumar', 'Chennai', '0.14', '007-22388644', '');
INSERT INTO AGENTS VALUES ('A012', 'Lucida', 'San Jose', '0.12', '044-52981425', '');
INSERT INTO AGENTS VALUES ('A005', 'Anderson', 'Brisban', '0.13', '045-21447739', '');
INSERT INTO AGENTS VALUES ('A001', 'Subbarao', 'Bangalore', '0.14', '077-12346674', '');
INSERT INTO AGENTS VALUES ('A002', 'Mukesh', 'Mumbai', '0.11', '029-12358964', '');
INSERT INTO AGENTS VALUES ('A006', 'McDen', 'London', '0.15', '078-22255588', '');
INSERT INTO AGENTS VALUES ('A004', 'Ivan', 'Torento', '0.15', '008-22544166', '');
INSERT INTO AGENTS VALUES ('A009', 'Benjamin', 'Hampshair', '0.11', '008-22536178', '');

SELECT * FROM AGENTS;

CREATE OR REPLACE TABLE  CUSTOMER
  (
   "CUST_CODE" VARCHAR2(6) NOT NULL PRIMARY KEY,
        "CUST_NAME" VARCHAR2(40) NOT NULL,
        "CUST_CITY" CHAR(35),
        "WORKING_AREA" VARCHAR2(35) NOT NULL,
        "CUST_COUNTRY" VARCHAR2(20) NOT NULL,
        "GRADE" NUMBER,
        "OPENING_AMT" NUMBER(12,2) NOT NULL,
        "RECEIVE_AMT" NUMBER(12,2) NOT NULL,
        "PAYMENT_AMT" NUMBER(12,2) NOT NULL,
        "OUTSTANDING_AMT" NUMBER(12,2) NOT NULL,
        "PHONE_NO" VARCHAR2(17) NOT NULL,
        "AGENT_CODE" CHAR(6) NOT NULL REFERENCES AGENTS
);
```

SQL SCRIP MATILLION SESSION 11

INSERT INTO CUSTOMER VALUES ('C00013', 'Holmes', 'London', 'London', 'UK', '2', '6000.00', '5000.00', '7000.00', '4000.00', 'BBBBBBB', 'A003');

INSERT INTO CUSTOMER VALUES ('C00001', 'Micheal', 'New York', 'New York', 'USA', '2', '3000.00', '5000.00', '2000.00', '6000.00', 'CCCCCCC', 'A008');

INSERT INTO CUSTOMER VALUES ('C00020', 'Albert', 'New York', 'New York', 'USA', '3', '5000.00', '7000.00', '6000.00', '6000.00', 'BBBBSBB', 'A008');

INSERT INTO CUSTOMER VALUES ('C00025', 'Ravindran', 'Bangalore', 'Bangalore', 'India', '2', '5000.00', '7000.00', '4000.00', '8000.00', 'AVAVAVA', 'A011');

INSERT INTO CUSTOMER VALUES ('C00024', 'Cook', 'London', 'London', 'UK', '2', '4000.00', '9000.00', '7000.00', '6000.00', 'FSDDSDF', 'A006');

INSERT INTO CUSTOMER VALUES ('C00015', 'Stuart', 'London', 'London', 'UK', '1', '6000.00', '8000.00', '3000.00', '11000.00', 'GFSGERS', 'A003');

INSERT INTO CUSTOMER VALUES ('C00002', 'Bolt', 'New York', 'New York', 'USA', '3', '5000.00', '7000.00', '9000.00', '3000.00', 'DDNRDRH', 'A008');

INSERT INTO CUSTOMER VALUES ('C00018', 'Fleming', 'Brisban', 'Brisban', 'Australia', '2', '7000.00', '7000.00', '9000.00', '5000.00', 'NHBGVFC', 'A005');

INSERT INTO CUSTOMER VALUES ('C00021', 'Jacks', 'Brisban', 'Brisban', 'Australia', '1', '7000.00', '7000.00', '7000.00', '7000.00', 'WERTGDF', 'A005');

INSERT INTO CUSTOMER VALUES ('C00019', 'Yearannaidu', 'Chennai', 'Chennai', 'India', '1', '8000.00', '7000.00', '7000.00', '8000.00', 'ZZZZBFV', 'A010');

INSERT INTO CUSTOMER VALUES ('C00005', 'Sasikant', 'Mumbai', 'Mumbai', 'India', '1', '7000.00', '11000.00', '7000.00', '11000.00', '147-25896312', 'A002');

INSERT INTO CUSTOMER VALUES ('C00007', 'Ramanathan', 'Chennai', 'Chennai', 'India', '1', '7000.00', '11000.00', '9000.00', '9000.00', 'GHRDWSD', 'A010');

INSERT INTO CUSTOMER VALUES ('C00022', 'Avinash', 'Mumbai', 'Mumbai', 'India', '2', '7000.00', '11000.00', '9000.00', '9000.00', '113-12345678','A002');

INSERT INTO CUSTOMER VALUES ('C00004', 'Winston', 'Brisban', 'Brisban', 'Australia', '1', '5000.00', '8000.00', '7000.00', '6000.00', 'AAAAAAA', 'A005');

INSERT INTO CUSTOMER VALUES ('C00023', 'Karl', 'London', 'London', 'UK', '0', '4000.00', '6000.00', '7000.00', '3000.00', 'AAAABAA', 'A006');

INSERT INTO CUSTOMER VALUES ('C00006', 'Shilton', 'Torento', 'Torento', 'Canada', '1', '10000.00', '7000.00', '6000.00', '11000.00', 'DDDDDDD', 'A004');

INSERT INTO CUSTOMER VALUES ('C00010', 'Charles', 'Hampshair', 'Hampshair', 'UK', '3', '6000.00', '4000.00', '5000.00', '5000.00', 'MMMMMMM', 'A009');

SQL SCRIP MATILLION SESSION 11

INSERT INTO CUSTOMER VALUES ('C00017', 'Srinivas', 'Bangalore', 'Bangalore', 'India', '2', '8000.00', '4000.00', '3000.00', '9000.00', 'AAAAAAB', 'A007');

INSERT INTO CUSTOMER VALUES ('C00012', 'Steven', 'San Jose', 'San Jose', 'USA', '1', '5000.00', '7000.00', '9000.00', '3000.00', 'KRFYGJK', 'A012');

INSERT INTO CUSTOMER VALUES ('C00008', 'Karolina', 'Torento', 'Torento', 'Canada', '1', '7000.00', '7000.00', '9000.00', '5000.00', 'HJKORED', 'A004');

INSERT INTO CUSTOMER VALUES ('C00003', 'Martin', 'Torento', 'Torento', 'Canada', '2', '8000.00', '7000.00', '7000.00', '8000.00', 'MJYURFD', 'A004');

INSERT INTO CUSTOMER VALUES ('C00009', 'Ramesh', 'Mumbai', 'Mumbai', 'India', '3', '8000.00', '7000.00', '3000.00', '12000.00', 'Phone No', 'A002');

INSERT INTO CUSTOMER VALUES ('C00014', 'Rangarappa', 'Bangalore', 'Bangalore', 'India', '2', '8000.00', '11000.00', '7000.00', '12000.00', 'AAAATGF', 'A001');

INSERT INTO CUSTOMER VALUES ('C00016', 'Venkatpati', 'Bangalore', 'Bangalore', 'India', '2', '8000.00', '11000.00', '7000.00', '12000.00', 'JRTVFDD', 'A007');

INSERT INTO CUSTOMER VALUES ('C00011', 'Sundariya', 'Chennai', 'Chennai', 'India', '3', '7000.00', '11000.00', '7000.00', '11000.00', 'PPHGRTS', 'A010');

SELECT * FROM CUSTOMER;

**********PIVOT/UNPIVOT**********;

/* You can use the PIVOT and UNPIVOT relational operators to change a table-valued expression into another table.

PIVOT rotates a table-valued expression by turning the unique values from one column in the expression into multiple columns in the output.

And PIVOT runs aggregations where they're required on any remaining column values that are wanted in the final output.

UNPIVOT carries out the opposite operation to PIVOT by rotating columns of a table-valued expression into column values. */

CREATE OR REPLACE TABLE PROD_PIVOT

(

    product_name VARCHAR(10),

SQL SCRIP MATILLION SESSION 11

```
    prod_details VARCHAR(20),
    storage NUMBER (5,0)
);

INSERT INTO PROD_PIVOT
VALUES('HP','RAM',8),
('HP','HardDisk',256),
('ACER','RAM',6),
('ACER','HardDisk',512),
('Lenovo','RAM',16),
('Lenovo','HardDisk',512);

select * from PROD_PIVOT;

create or replace table data(id int, a varchar(10), b varchar(10), c varchar(10));
insert into data(id,a,b,c) values(1,'a1','b1','c1'),(2,'a2','b2','c2');
select * from data;

--Here's the query to do unpivot in SQL.
--Since MySQL doesn't offer an UNPIVOT function,
--You need to use UNION ALL clause in to reverse pivot a table in MySQL.

select id, 'a' col, a value
    from data
    union all
    select id, 'b' col, b value
    from data
    union all
    select id, 'c' col, c value
```

SQL SCRIP MATILLION SESSION 11

```
    from data;


--In the above query, we basically cut the original table into 3 smaller ones – one for each column a,b,c

--and then append them one below the other using UNION ALL.


-- How to Transpose Rows to Columns Dynamically in MySQL

--Here's how to create dynamic pivot tables in MySQL. Let's say you have the following table

CREATE OR REPLACE TABLE Meeting
(
   ID INT,
   Meeting_id INT,
   field_key VARCHAR(100),
   field_value VARCHAR(100)
);


INSERT INTO Meeting(ID,Meeting_id,field_key,field_value)
VALUES (1, 1,'first_name' , 'Alec');
INSERT INTO Meeting(ID,Meeting_id,field_key,field_value)
VALUES (2, 1,'last_name' , 'Jones');
INSERT INTO Meeting(ID,Meeting_id,field_key,field_value)
VALUES (3, 1,'occupation' , 'engineer');
INSERT INTO Meeting(ID,Meeting_id,field_key,field_value)
VALUES (4,2,'first_name' , 'John');
INSERT INTO Meeting(ID,Meeting_id,field_key,field_value)
VALUES (5,2,'last_name' , 'Doe');
INSERT INTO Meeting(ID,Meeting_id,field_key,field_value)
VALUES (6,2,'occupation' , 'engineer');
```

SQL SCRIP MATILLION SESSION 11

```
SELECT * FROM MEETING;


--Let's say you want to transpose rows to columns dynamically,

-- such that a new column is created for each unique value in field_key column, that is (first_name, last_name, occupation)



CREATE OR REPLACE TABLE TRANSACTION_RAW

(

trans_id INT,

account_id      INT,

txn_date        DATE,

txn_type        VARCHAR(30),

operation       VARCHAR(40),

amount INT,

balance FLOAT,

Purpose         VARCHAR(40),

bank    VARCHAR(45),

account_type INT

--FOREIGN KEY (account_id) references ACCOUNT(account_id)

);


select * from TRANSACTION_RAW;


update TRANSACTION_RAW

set bank = 'DBS' WHERE bank is null;


 SELECT * ,

 SPLIT_PART(TXN_DATE, '-', 1) AS "year",
```

SQL SCRIP MATILLION SESSION 11

```
 SPLIT_PART(TXN_DATE, '-', 2) AS "month",
 SPLIT_PART(TXN_DATE, '-', 3) AS "day"
FROM TRANSACTION_RAW  ;
```

```
CREATE OR REPLACE TABLE STUDENT
(
   stud_id NUMBER(1,0) NOT NULL,
   phy_marks NUMBER(3,1),
   chem_marks NUMBER(3,1),
   maths_marks NUMBER(3,1)
);

INSERT INTO STUDENT
VALUES(1,88.5,73.5,95.5),(2,80,85,90),(3,75,67.5,91.5),(4,72.5,85.5,93.5),(5,72,82,90);

SELECT * FROM STUDENT;
```