MKJ IT Learnings

The Corporate & Online Training Company
www.mkj-it-learnings.com

RDBMS

Storing Data into Database

Created By : Ashish Bansal

Session Outline

1. DDL (Data Definition Language) Commands
2. DML (Data Manipulation Language) Commands
3. Select Operations.
4. Inbuild Functions.
5. Stored Procedures
6. Triggers
7. PL-SQL

Relational Database

- Data stored in rows and columns
- Values are atomic
- Columns are undistinguished and having unique names.
- Data is in sequence.
- Support SQL for quiring data.
- Focused on Normalization
  (normalization is the process to store the data to avoid duplication and redundancy).
  - To minimize duplication of data.
  - To minimize or avoid data modification issues.

First Normal Form

First normal form focus on primary key

1. Each set of column must have unique value.
2. Each row should have a primary key .

Student Name	Course	Age
Ramesh	Java, Salesforce	24
Rakesh	Excel	26
Lokesh	Spring Cloud	31

$\longrightarrow$

Student Name	Course	Age
Ramesh	Java	24
Ramesh	Salesforce	24
Rakesh	Excel	26
Lokesh	Spring Cloud	31

Second Normal Form-

1. Table should be in 1NF.

2. There should not be any partial dependency on Primary key.

Student Name	Age
Ramesh	24
Rakesh	26
Lokesh	31

Student Name	Course
Ramesh	Java
Rakesh	Excel
Lokesh	Spring Cloud
Ramesh	Salesforce

Creation of Table

Create table is the command which includes , information of attributes along with data type and constraints .

Constraints

1) Primary Key
2) Not Null
3) Unique
4) Foreign Key
5) Check Constraints. (used for better data validation purpose)
6) Default
7) Index

Syntax of Creation of Table

```
CREATE TABLE table_name (
    column1 datatype constraint,
    column2 datatype constraint,
    column3 datatype constraint,
    ....
);
```

Creation of Table

```
create table Instructor(
          instructorCode int Primary Key,
           name varchar(20) not null,
           salary int not null,
           jobStartYear int not null
);
```

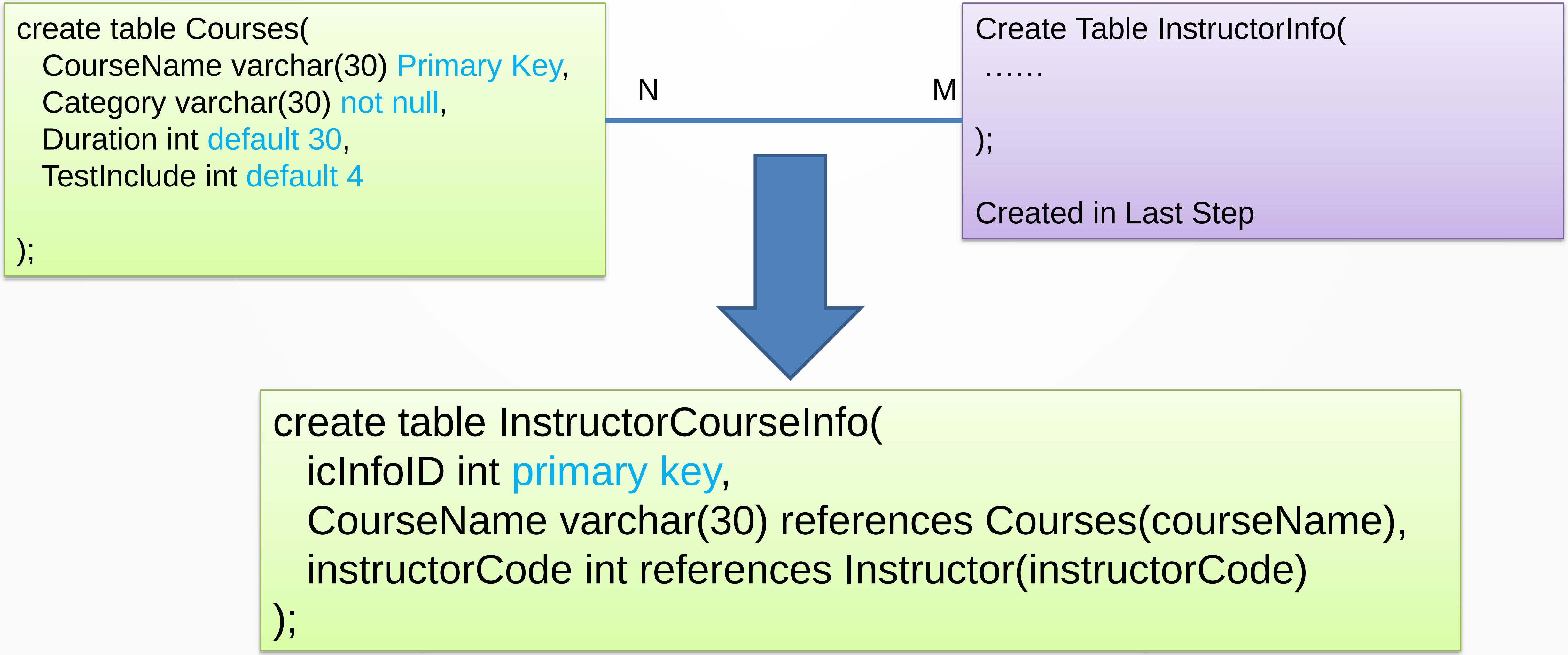
1

1. createtable.txt

1

```
Create Table InstructorInfo(

recordId int primary key,
instructorCode int references Instructor(instructorCode),
address varchar(100),
email varchar(30),
phone number(10),
firstLanguage varchar(20),
secondLanguage Varchar(20)

);
```

Continue…



```
create table Courses(
   CourseName varchar(30) Primary Key,
   Category varchar(30) not null,
   Duration int default 30,
   TestInclude int default 4

);
```

```
Create Table InstructorInfo(
 ……

);

Created in Last Step
```

```
create table InstructorCourseInfo(
   icInfoID int primary key,
   CourseName varchar(30) references Courses(courseName),
   instructorCode int references Instructor(instructorCode)
);
```

Note :- Go Through Student and Batch Table Creation Script in Script File.

Alter Table and Drop Table

```
ALTER TABLE "table_name"
ADD "column_name" "Data Type";
```

CURD Operations

```
INSERT INTO "table_name" ("column1", "column2", ...)  VALUES ("value1", "value2", ...);
```

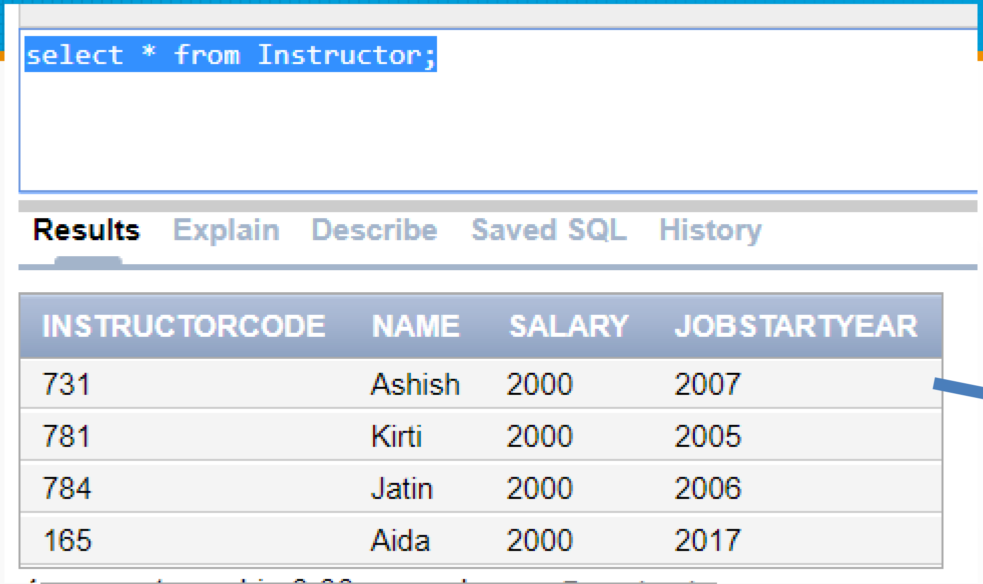
```
UPDATE "table_name"
SET "column_1" = [new value]
WHERE "condition";
```

Select statements

```
DELETE  from "table_name"
WHERE "condition";
```

Insert Query

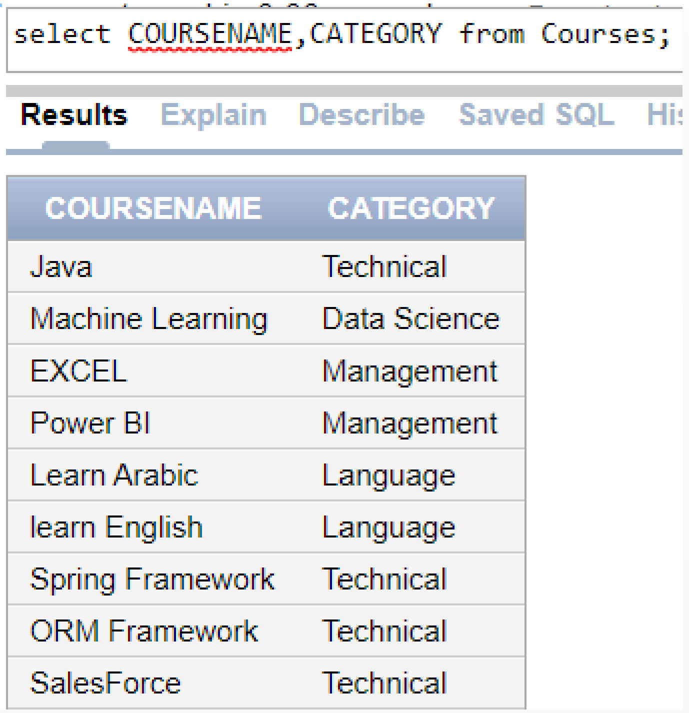
2. InsertScript.txt



```
select * from Instructor;
```

Results Explain Describe Saved SQL History

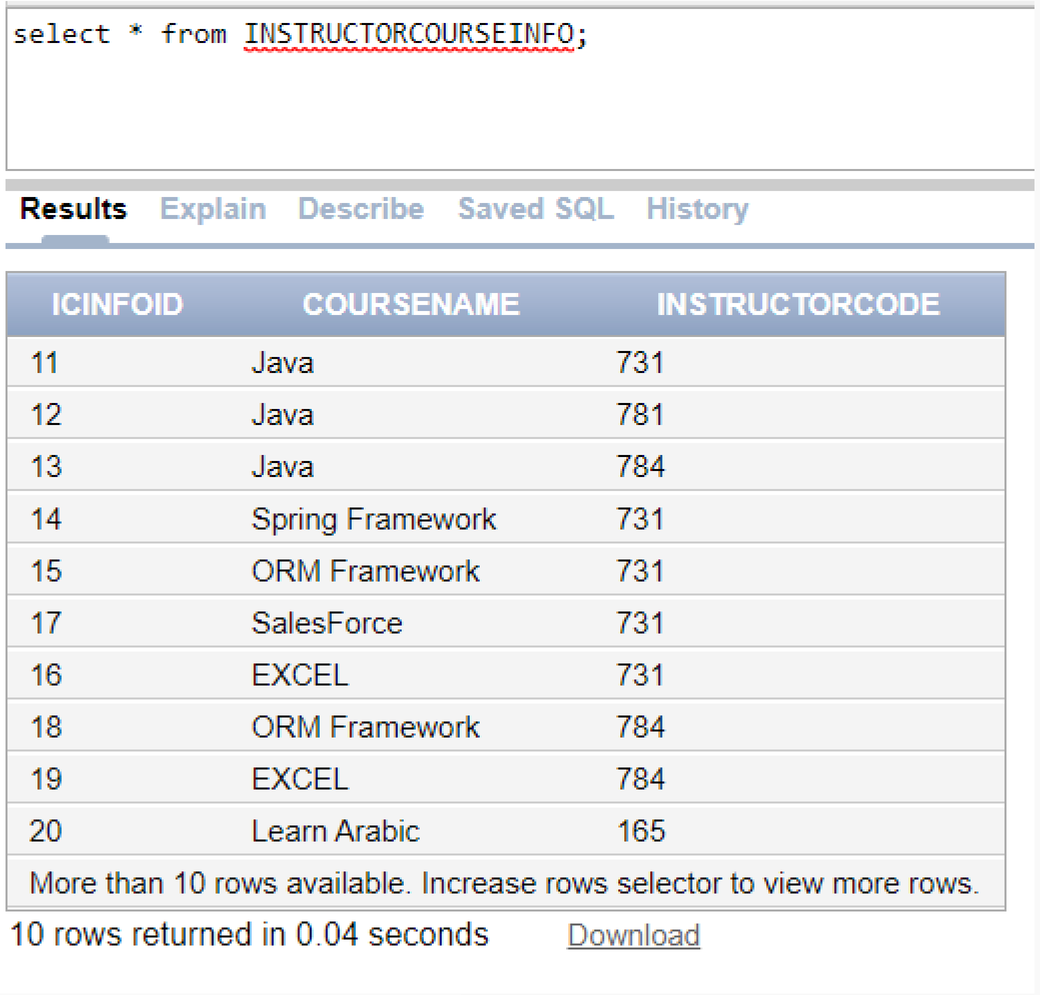
INSTRUCTORCODE	NAME	SALARY	JOBSTARTYEAR
731	Ashish	2000	2007
781	Kirti	2000	2005
784	Jatin	2000	2006
165	Aida	2000	2017



```
select COURSENAME,CATEGORY from Courses;
```

Results Explain Describe Saved SQL His

COURSENAME	CATEGORY
Java	Technical
Machine Learning	Data Science
EXCEL	Management
Power BI	Management
Learn Arabic	Language
learn English	Language
Spring Framework	Technical
ORM Framework	Technical
SalesForce	Technical



```
select * from INSTRUCTORCOURSEINFO;
```

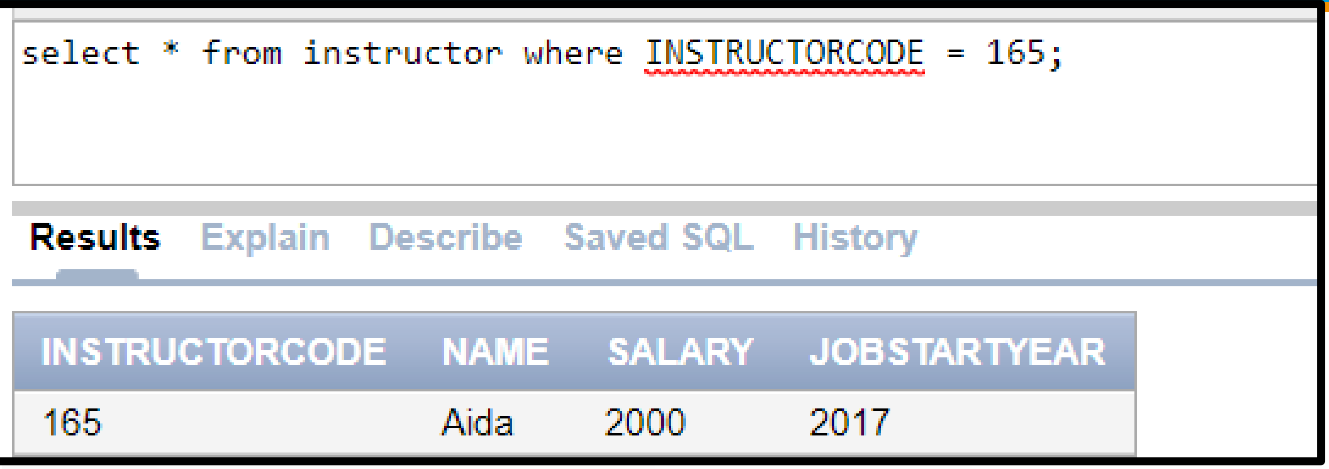
Results Explain Describe Saved SQL History

ICINFOID	COURSENAME	INSTRUCTORCODE
11	Java	731
12	Java	781
13	Java	784
14	Spring Framework	731
15	ORM Framework	731
17	SalesForce	731
16	EXCEL	731
18	ORM Framework	784
19	EXCEL	784
20	Learn Arabic	165

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.04 seconds Download

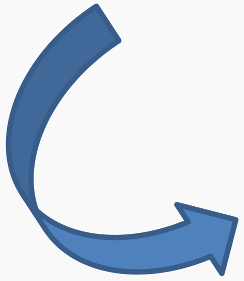
Update Query

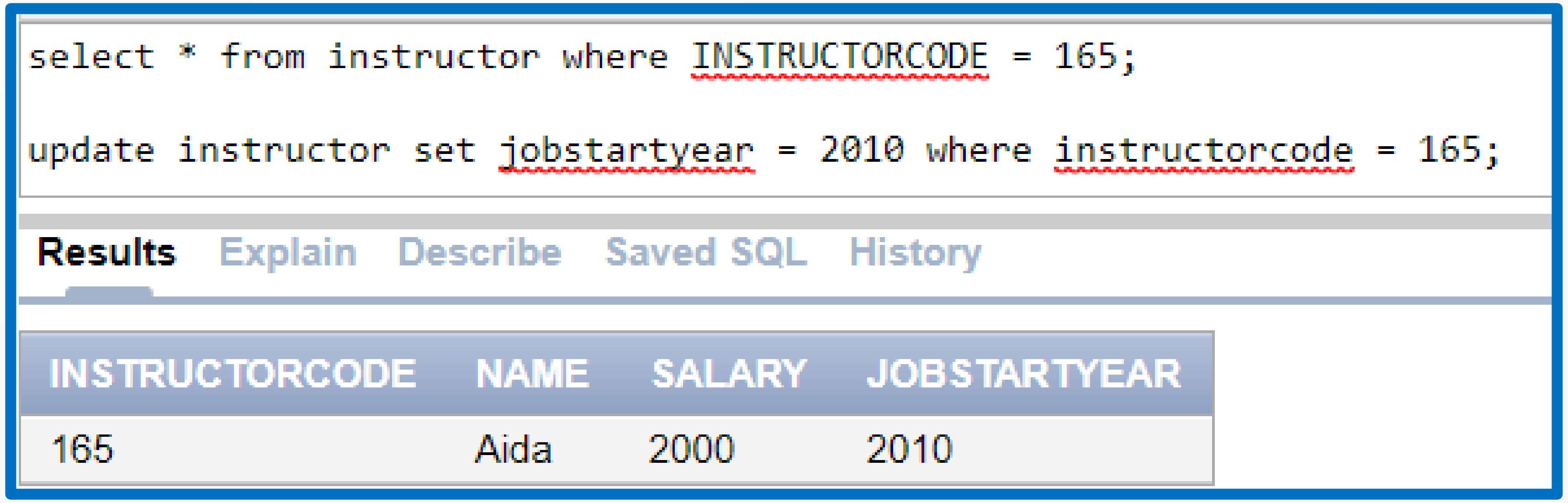


```
select * from instructor where INSTRUCTORCODE = 165;
```

INSTRUCTORCODE	NAME	SALARY	JOBSTARTYEAR
165	Aida	2000	2017

update instructor set jobstartyear = 2010 where instructorcode = 165;





```
select * from instructor where INSTRUCTORCODE = 165;

update instructor set jobstartyear = 2010 where instructorcode = 165;
```

INSTRUCTORCODE	NAME	SALARY	JOBSTARTYEAR
165	Aida	2000	2010

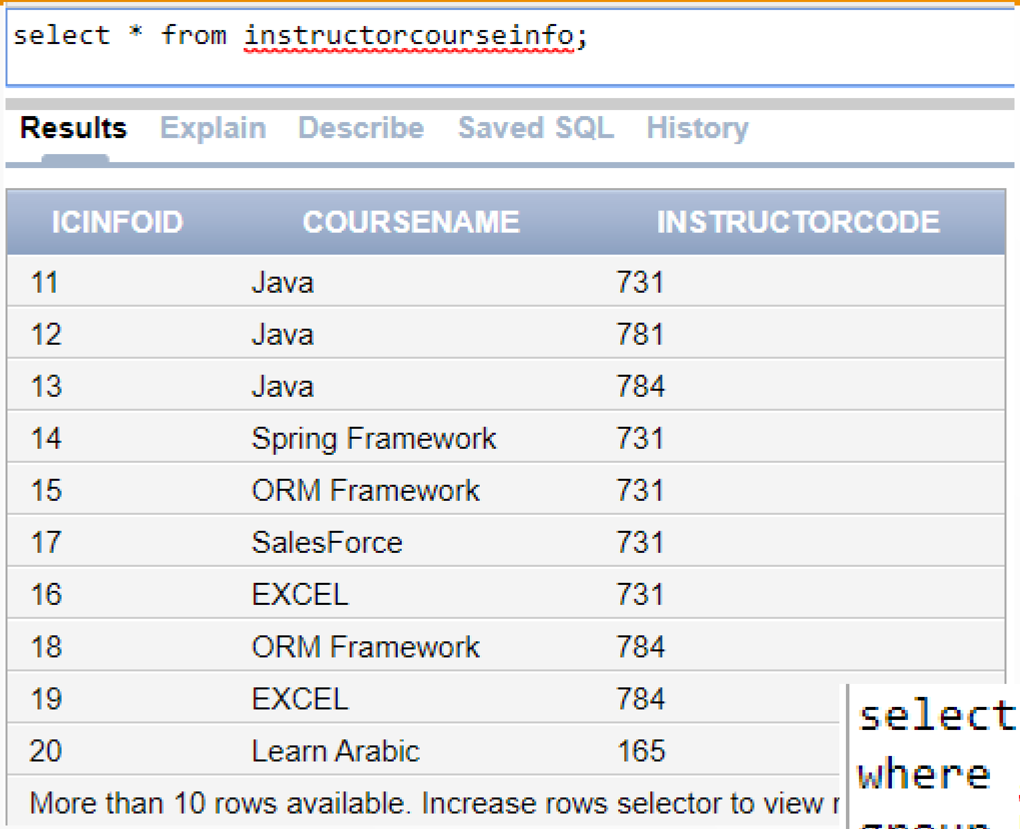
Select Statements

1) Where
2) In
3) Not In
4) Between
5) AND & OR
6) LIKE
7) ORDER BY
8) GROUP BY
9) HAVING
10) UNIQUE

SQL Functions

1) Average
2) Count
3) MAX
4) MIN
5) SUM
6) ROUND

Select Statements – Group By

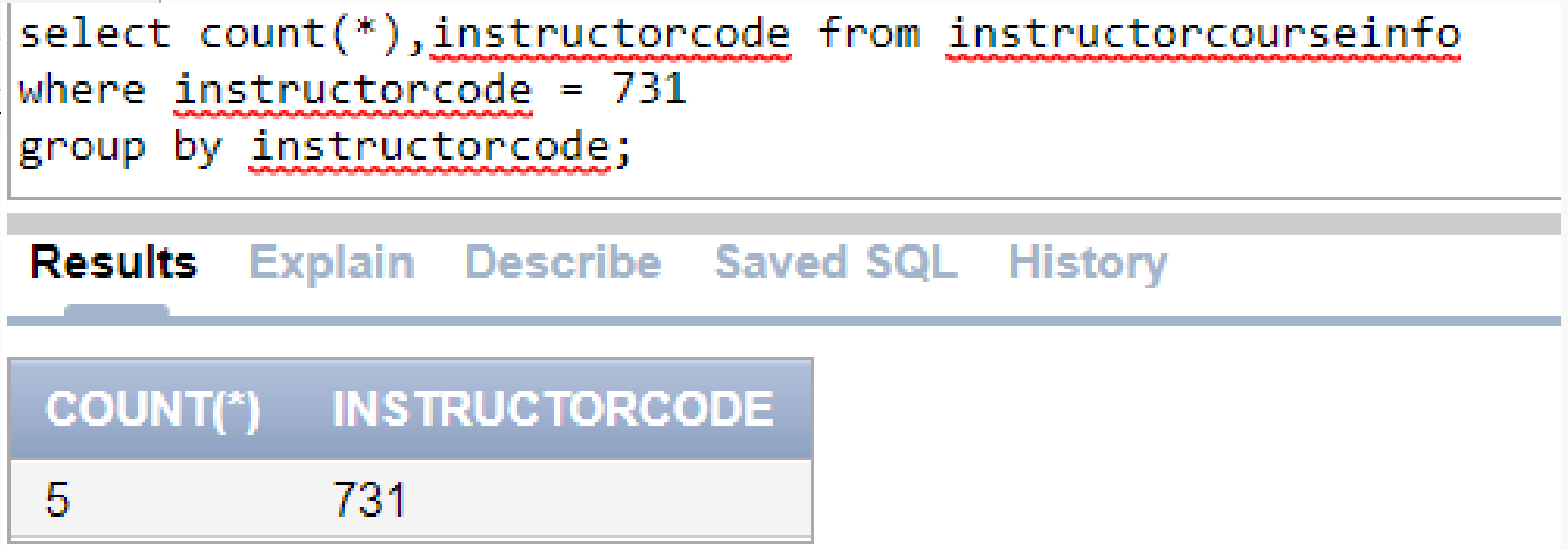


```
select * from instructorcourseinfo;
```

ICINFOID	COURSENAME	INSTRUCTORCODE
11	Java	731
12	Java	781
13	Java	784
14	Spring Framework	731
15	ORM Framework	731
17	SalesForce	731
16	EXCEL	731
18	ORM Framework	784
19	EXCEL	784
20	Learn Arabic	165

More than 10 rows available. Increase rows selector to view r

Objective : To find how many courses handled by trainer number 731

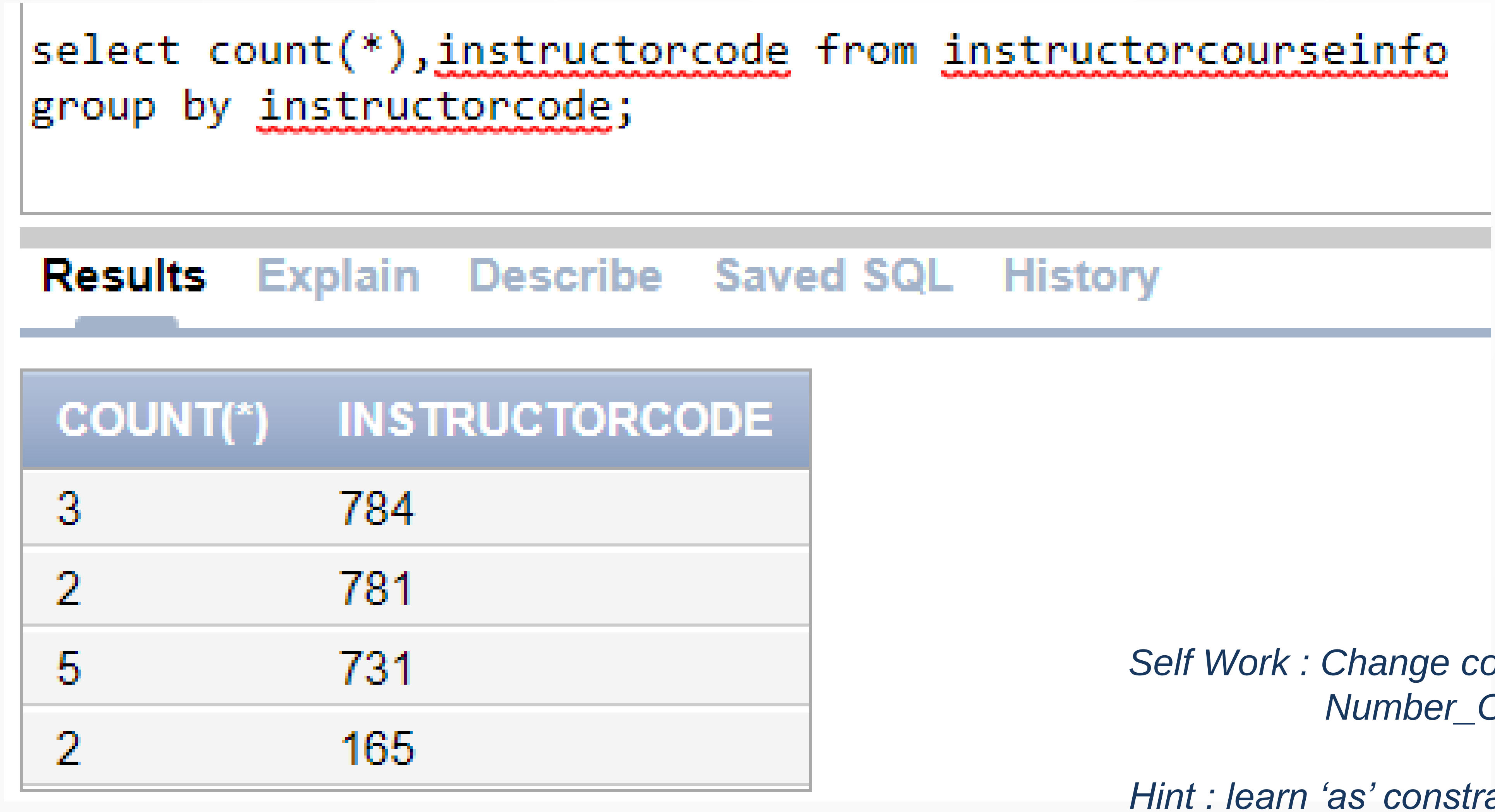


```
select count(*),instructorcode from instructorcourseinfo
where instructorcode = 731
group by instructorcode;
```

COUNT(*)	INSTRUCTORCODE
5	731

Select Statements – Group By Continue…

Objective : How many courses handled by trainers



```
select count(*),instructorcode from instructorcourseinfo
group by instructorcode;
```

Results Explain Describe Saved SQL History

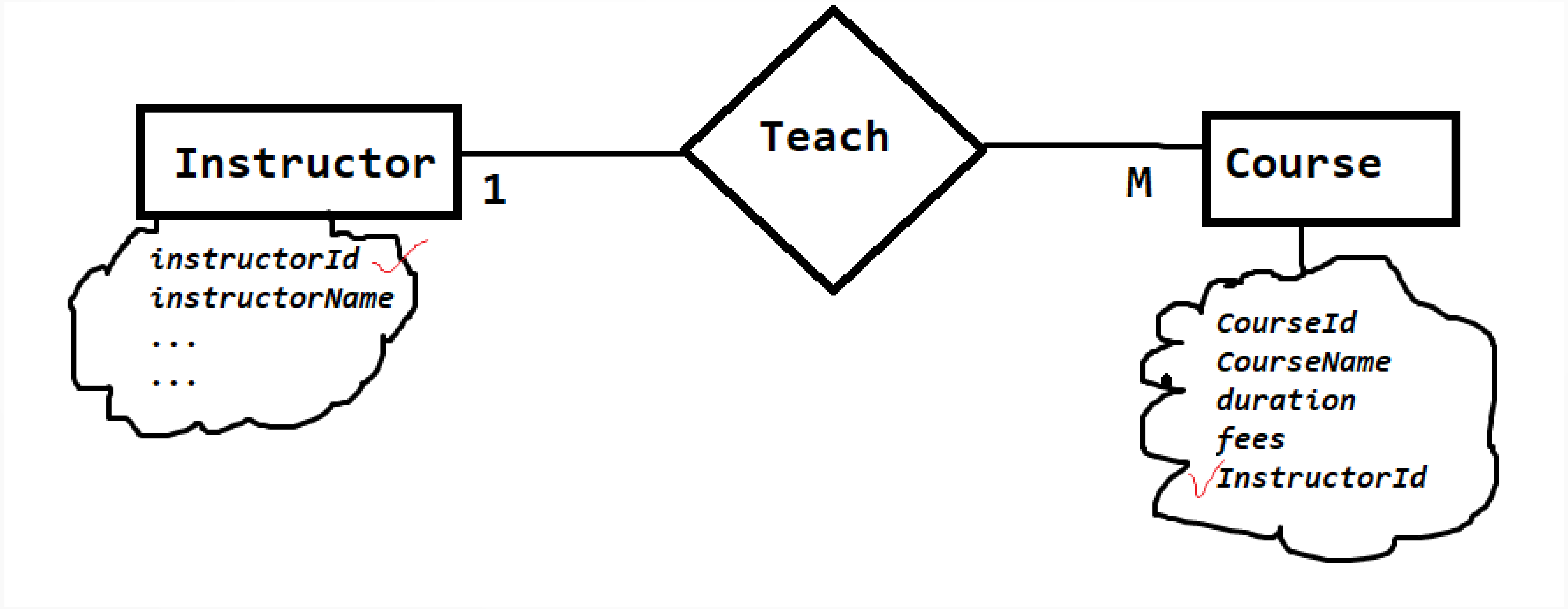
COUNT(*)	INSTRUCTORCODE
3	784
2	781
5	731
2	165

Self Work : Change column name count(*) to Number_Of_Courses

Hint : learn 'as' constraint

Joins

A relational database consists of multiple related tables linking together using common columns, which are known as foreign key columns. Because of this, data in each table is incomplete from the business perspective.



*To get complete information, we have to query data from both **Instructor & Course** tables.*
A join is a method of linking data between one or more tables based on values of the common column between the tables.

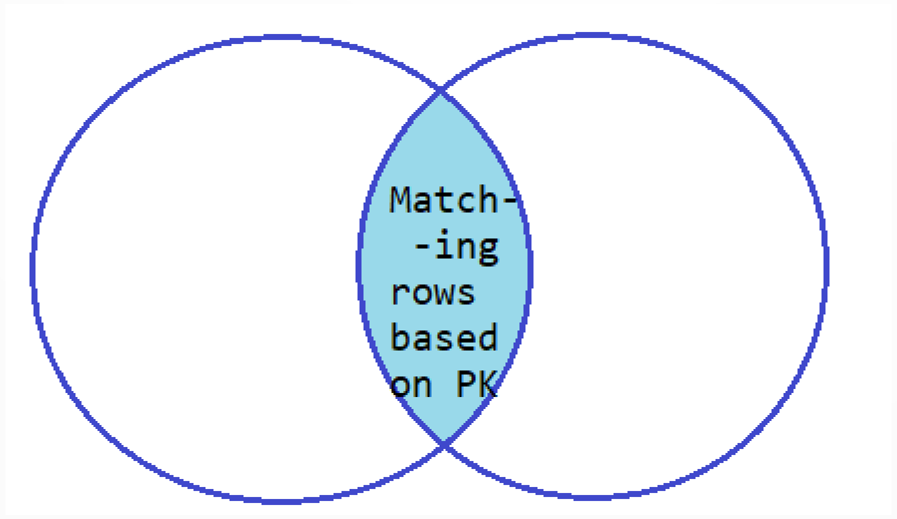
Terminology

What is Left and Right Table ?

- ✓ Left Table : - The Table which used to make select statement.
- ✓Right Table :- The Table which used for joining left table

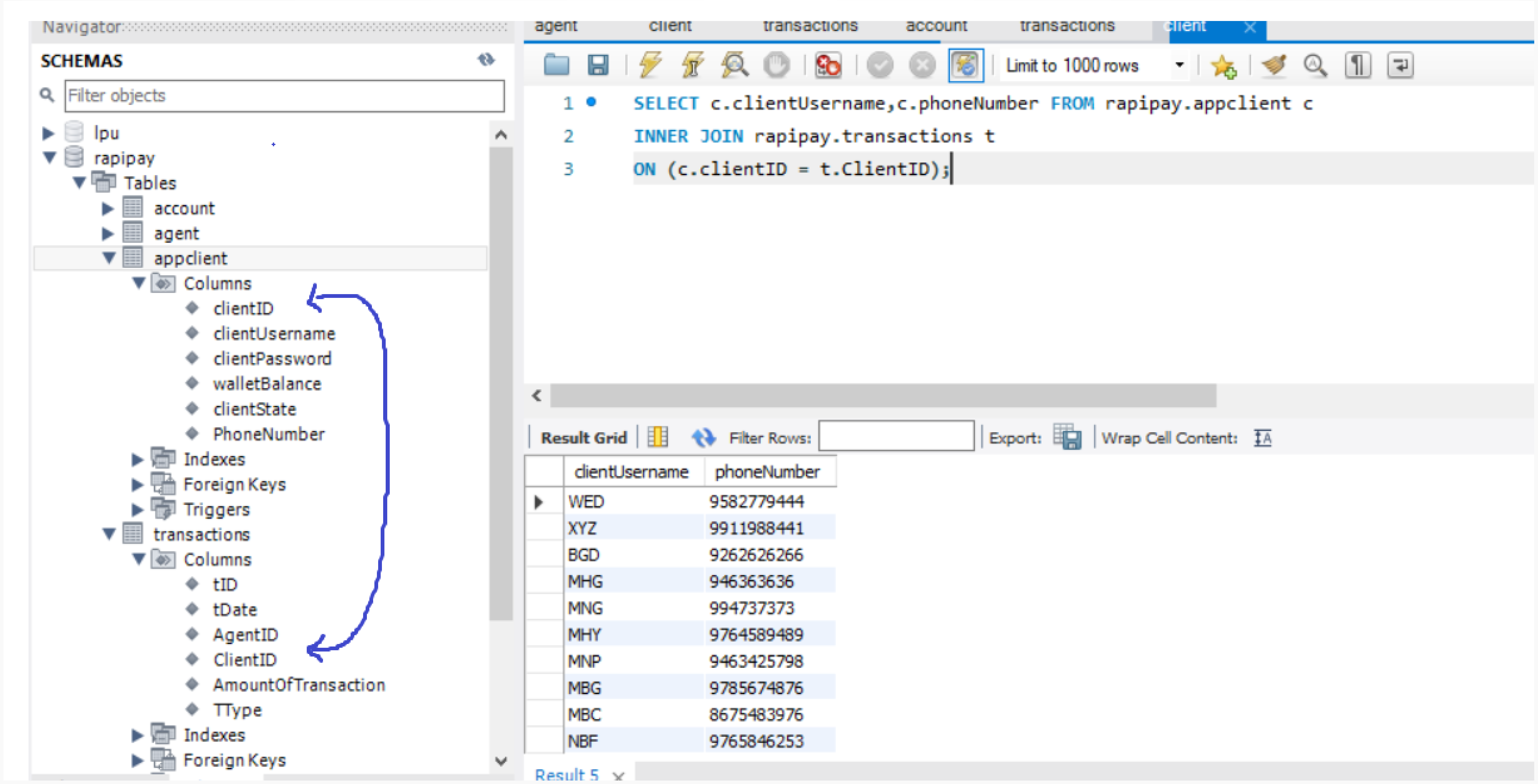
Where to Use Inner Join

It produces the data set that includes rows from the left join which have matching rows from the right tables



```
Select * from Employee
Inner Join EmployeeDetails ON
Employee.empID = EmployeeDetails.empID
```

Inner Join

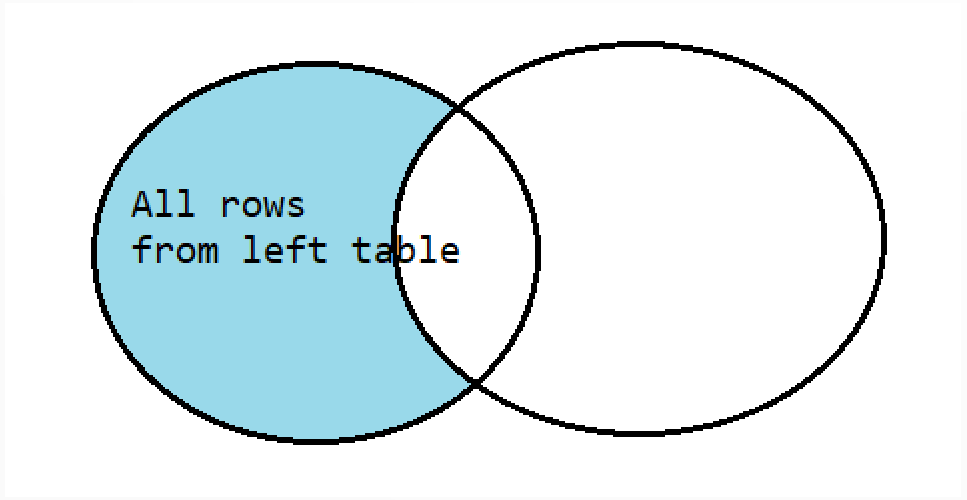


```
SELECT c.clientUsername,c.phoneNumber FROM rapipay.appclient c
INNER JOIN rapipay.transactions t
ON (c.clientID = t.ClientID);
```

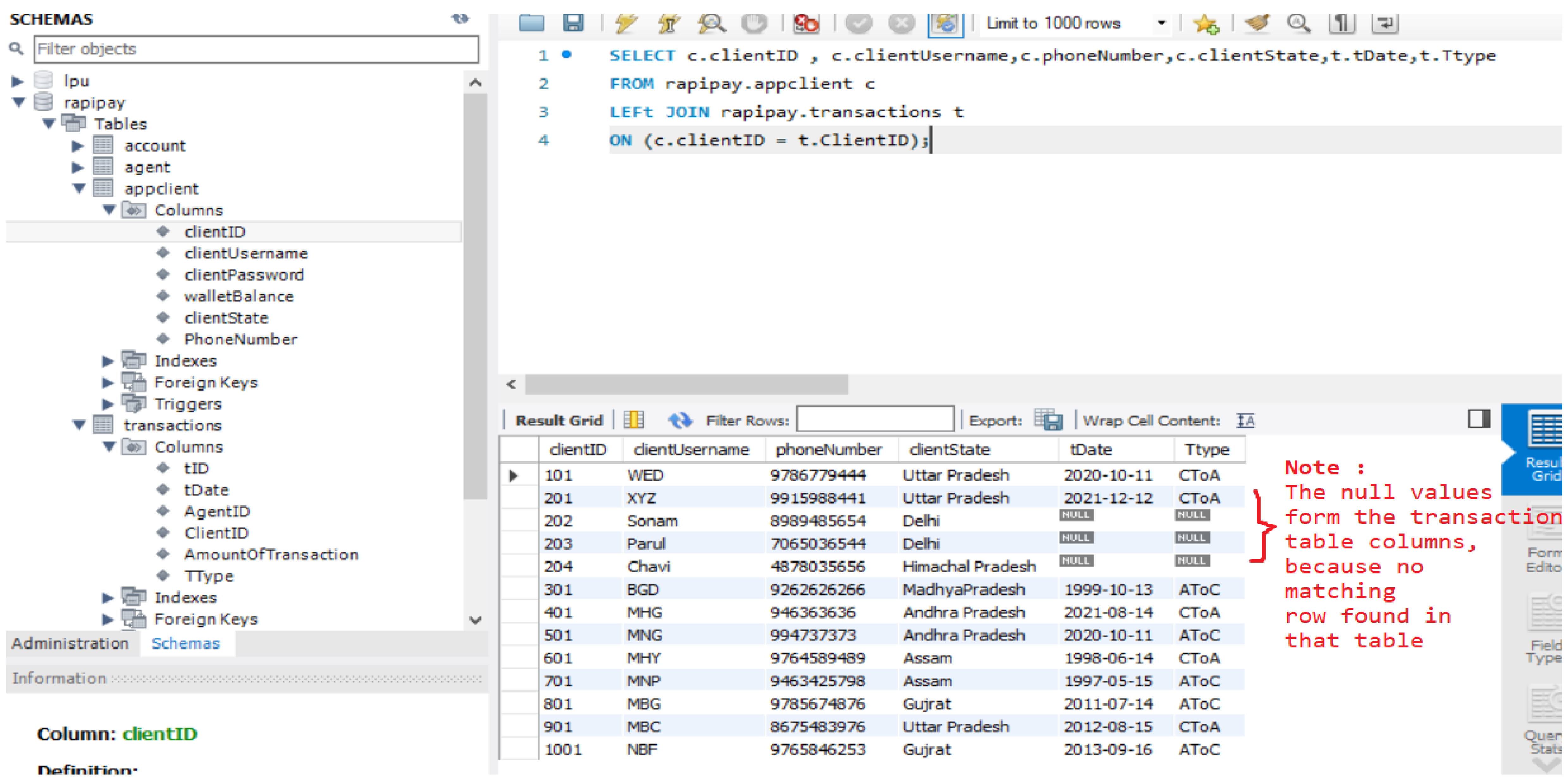
clientUsername	phoneNumber
WED	9582779444
XYZ	9911988441
BGD	9262626266
MHG	946363636
MNG	994737373
MHY	9764589489
MNP	9463425798
MBG	9785674876
MBC	8675483976
NBF	9765846253

Left Join

- Return all records from the left table , even if there are no records matches in the right table.
- For unmatched values in the right tables null is displayed.



Left Join

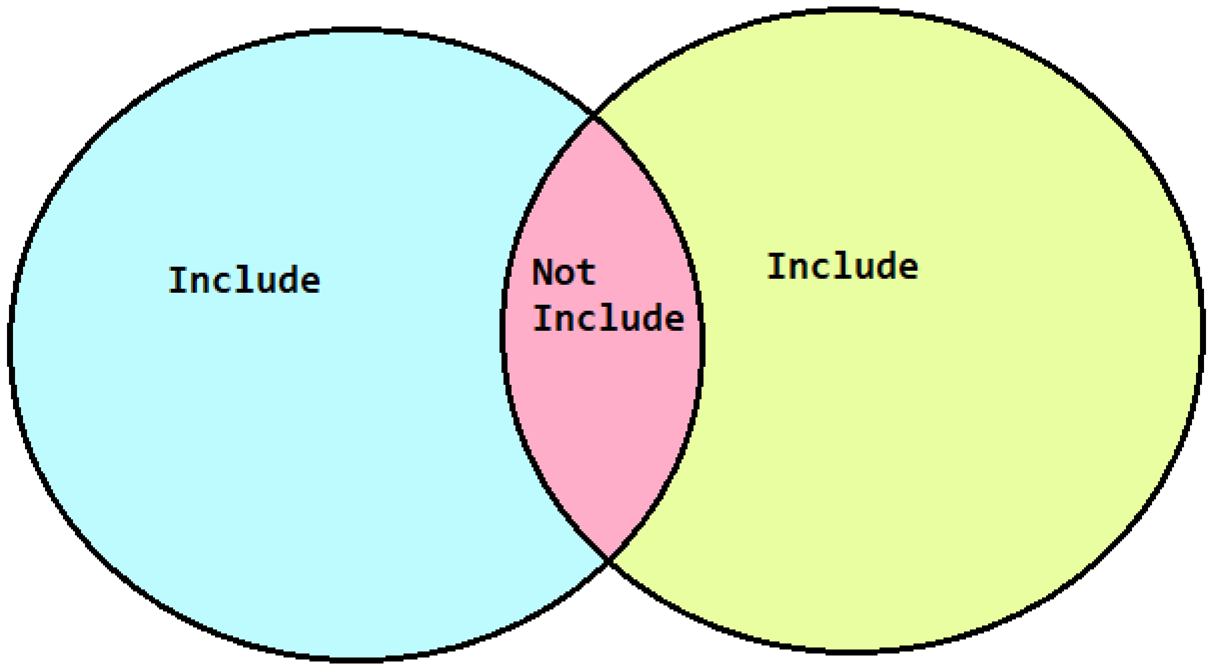


```sql
SELECT c.clientID , c.clientUsername,c.phoneNumber,c.clientState,t.tDate,t.Ttype
FROM rapipay.appclient c
LEFt JOIN rapipay.transactions t
ON (c.clientID = t.ClientID);
```

clientID	clientUsername	phoneNumber	clientState	tDate	Ttype
101	WED	9786779444	Uttar Pradesh	2020-10-11	CToA
201	XYZ	9915988441	Uttar Pradesh	2021-12-12	CToA
202	Sonam	8989485654	Delhi	NULL	NULL
203	Parul	7065036544	Delhi	NULL	NULL
204	Chavi	4878035656	Himachal Pradesh	NULL	NULL
301	BGD	9262626266	MadhyaPradesh	1999-10-13	AToC
401	MHG	946363636	Andhra Pradesh	2021-08-14	CToA
501	MNG	994737373	Andhra Pradesh	2020-10-11	AToC
601	MHY	9764589489	Assam	1998-06-14	CToA
701	MNP	9463425798	Assam	1997-05-15	AToC
801	MBG	9785674876	Gujrat	2011-07-14	AToC
901	MBC	8675483976	Uttar Pradesh	2012-08-15	CToA
1001	NBF	9765846253	Gujrat	2013-09-16	AToC

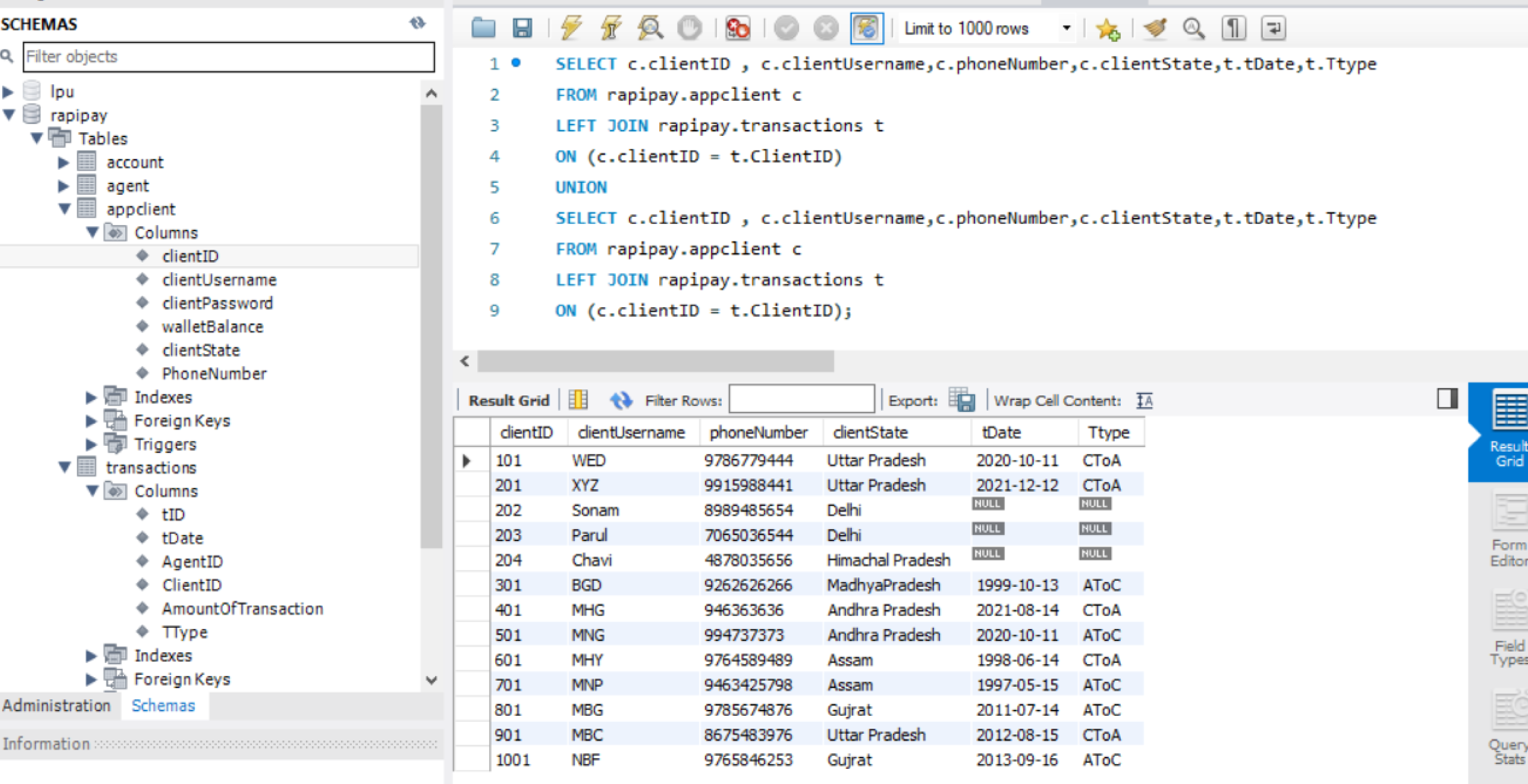
Note :
The null values form the transaction table columns, because no matching row found in that table

SQL Full Join



Full Join = (LEFT JOIN + RIGHT JOIN)- DUPLICATE Records

SQL Full Join (Union Clause in MYSQL)



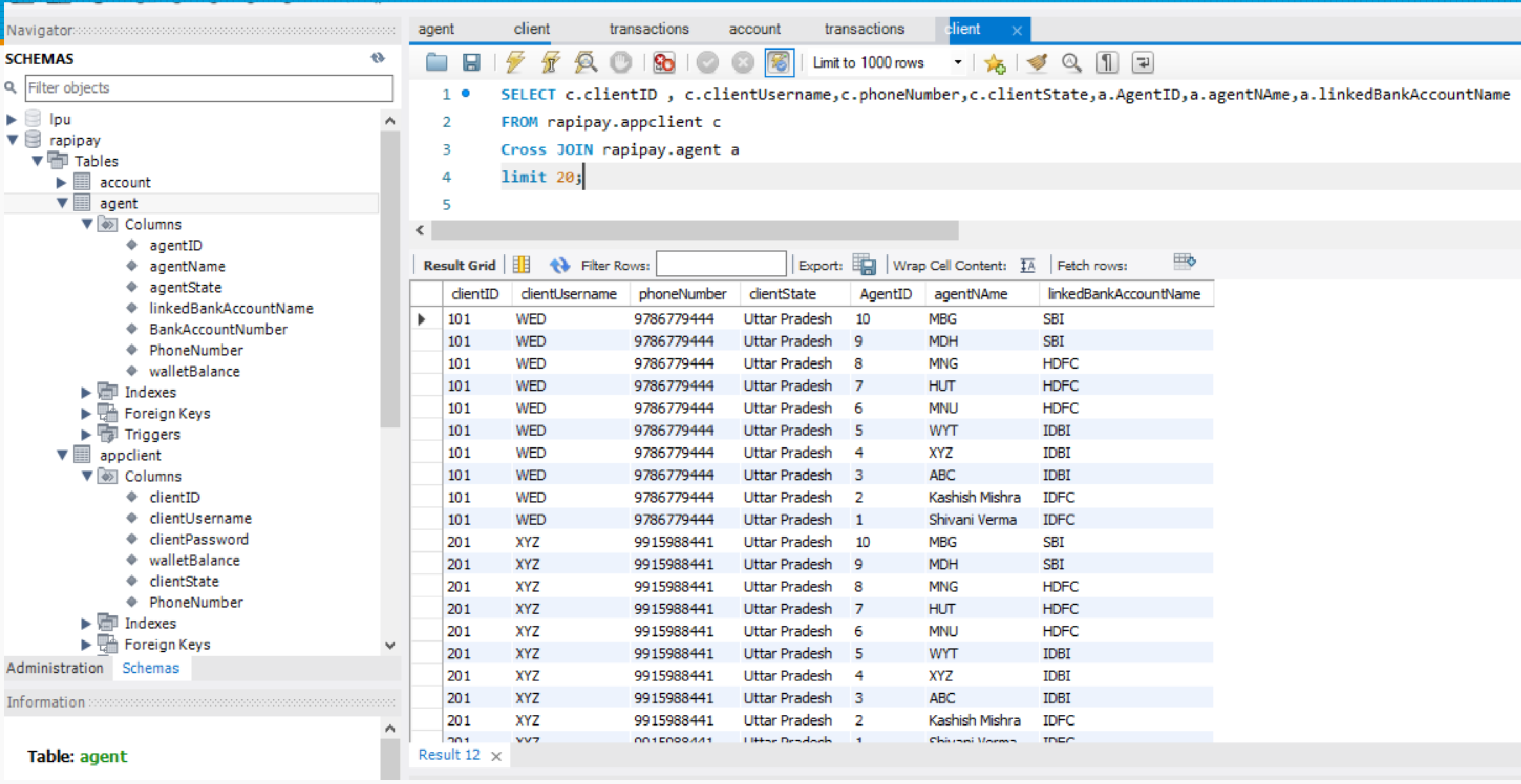
```
SELECT c.clientID , c.clientUsername,c.phoneNumber,c.clientState,t.tDate,t.Ttype
FROM rapipay.appclient c
LEFT JOIN rapipay.transactions t
ON (c.clientID = t.ClientID)
UNION
SELECT c.clientID , c.clientUsername,c.phoneNumber,c.clientState,t.tDate,t.Ttype
FROM rapipay.appclient c
LEFT JOIN rapipay.transactions t
ON (c.clientID = t.ClientID);
```

clientID	clientUsername	phoneNumber	clientState	tDate	Ttype
101	WED	9786779444	Uttar Pradesh	2020-10-11	CToA
201	XYZ	9915988441	Uttar Pradesh	2021-12-12	CToA
202	Sonam	8989485654	Delhi	NULL	NULL
203	Parul	7065036544	Delhi	NULL	NULL
204	Chavi	4878035656	Himachal Pradesh	NULL	NULL
301	BGD	9262626266	MadhyaPradesh	1999-10-13	AToC
401	MHG	946363636	Andhra Pradesh	2021-08-14	CToA
501	MNG	994737373	Andhra Pradesh	2020-10-11	AToC
601	MHY	9764589489	Assam	1998-06-14	CToA
701	MNP	9463425798	Assam	1997-05-15	AToC
801	MBG	9785674876	Gujrat	2011-07-14	AToC
901	MBC	8675483976	Uttar Pradesh	2012-08-15	CToA
1001	NBF	9765846253	Gujrat	2013-09-16	AToC

Cross Join

- Every record of left table associate with right table
- No need of any common column between tables.

Cross Join



```
SELECT c.clientID , c.clientUsername,c.phoneNumber,c.clientState,a.AgentID,a.agentNAme,a.linkedBankAccountName
FROM rapipay.appclient c
Cross JOIN rapipay.agent a
limit 20;
```

clientID	clientUsername	phoneNumber	clientState	AgentID	agentNAme	linkedBankAccountName
101	WED	9786779444	Uttar Pradesh	10	MBG	SBI
101	WED	9786779444	Uttar Pradesh	9	MDH	SBI
101	WED	9786779444	Uttar Pradesh	8	MNG	HDFC
101	WED	9786779444	Uttar Pradesh	7	HUT	HDFC
101	WED	9786779444	Uttar Pradesh	6	MNU	HDFC
101	WED	9786779444	Uttar Pradesh	5	WYT	IDBI
101	WED	9786779444	Uttar Pradesh	4	XYZ	IDBI
101	WED	9786779444	Uttar Pradesh	3	ABC	IDBI
101	WED	9786779444	Uttar Pradesh	2	Kashish Mishra	IDFC
101	WED	9786779444	Uttar Pradesh	1	Shivani Verma	IDFC
201	XYZ	9915988441	Uttar Pradesh	10	MBG	SBI
201	XYZ	9915988441	Uttar Pradesh	9	MDH	SBI
201	XYZ	9915988441	Uttar Pradesh	8	MNG	HDFC
201	XYZ	9915988441	Uttar Pradesh	7	HUT	HDFC
201	XYZ	9915988441	Uttar Pradesh	6	MNU	HDFC
201	XYZ	9915988441	Uttar Pradesh	5	WYT	IDBI
201	XYZ	9915988441	Uttar Pradesh	4	XYZ	IDBI
201	XYZ	9915988441	Uttar Pradesh	3	ABC	IDBI
201	XYZ	9915988441	Uttar Pradesh	2	Kashish Mishra	IDFC
201	XYZ	9915988441	Uttar Pradesh	1	Shivani Verma	IDFC

Stored Procedure

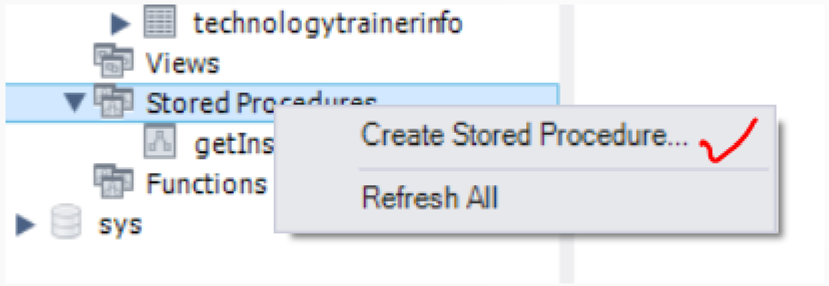
*RDBMS databases has **the feature called stored procedure**, which is a set of SQL statements which can perform repetitive task. The stored procedure can be called using applications like **Java** ... or other stored procedures.*

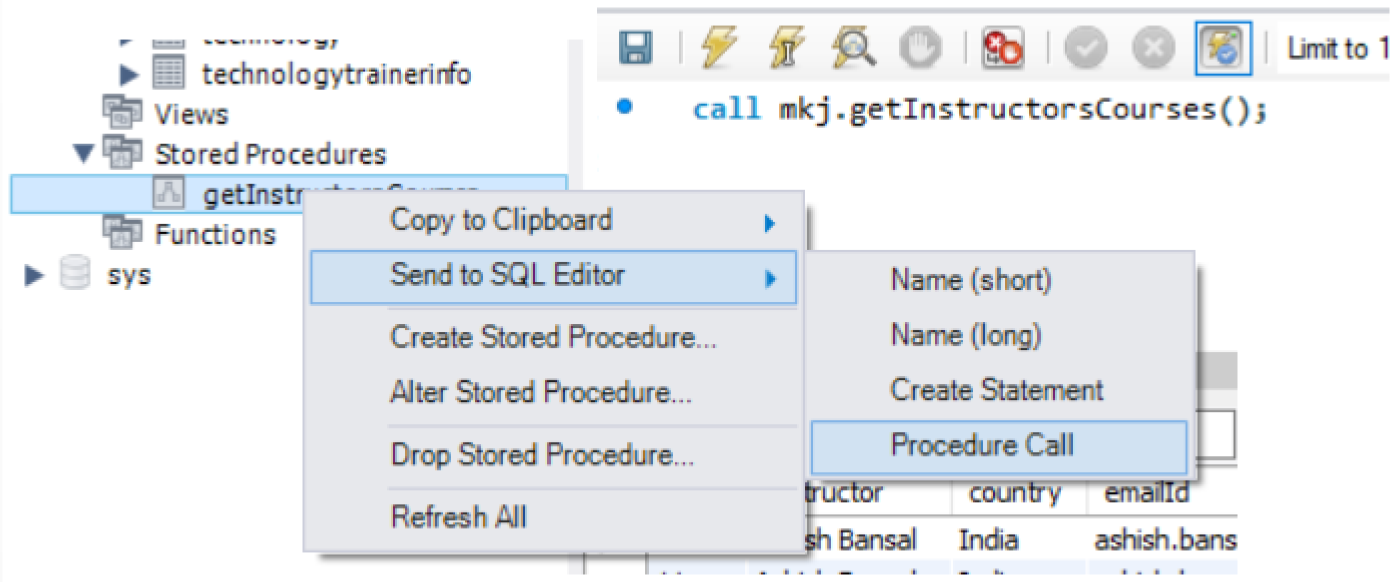
Stored procedure limits the user's direct access to data tables. It gives an interface and secure mechanism to manipulate data. With stored procedures, the repetition of code is avoided and the code block can be called whenever necessary. It allows variable declaration, parameter passing; return multiple values, flow statements.

Stored procedures are created using CREATE PROCEDURE

Stored Procedure- Creation & Calling

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `getInstructorsCourses`()
BEGIN
select i.instructorId as id, i.instructorName as Instructor , i.country,i.emailId,c.courseName as course,c.fees
from mkj.instructor as i
inner join mkj.course c on c.instructorId = i.instructorId;
END
```

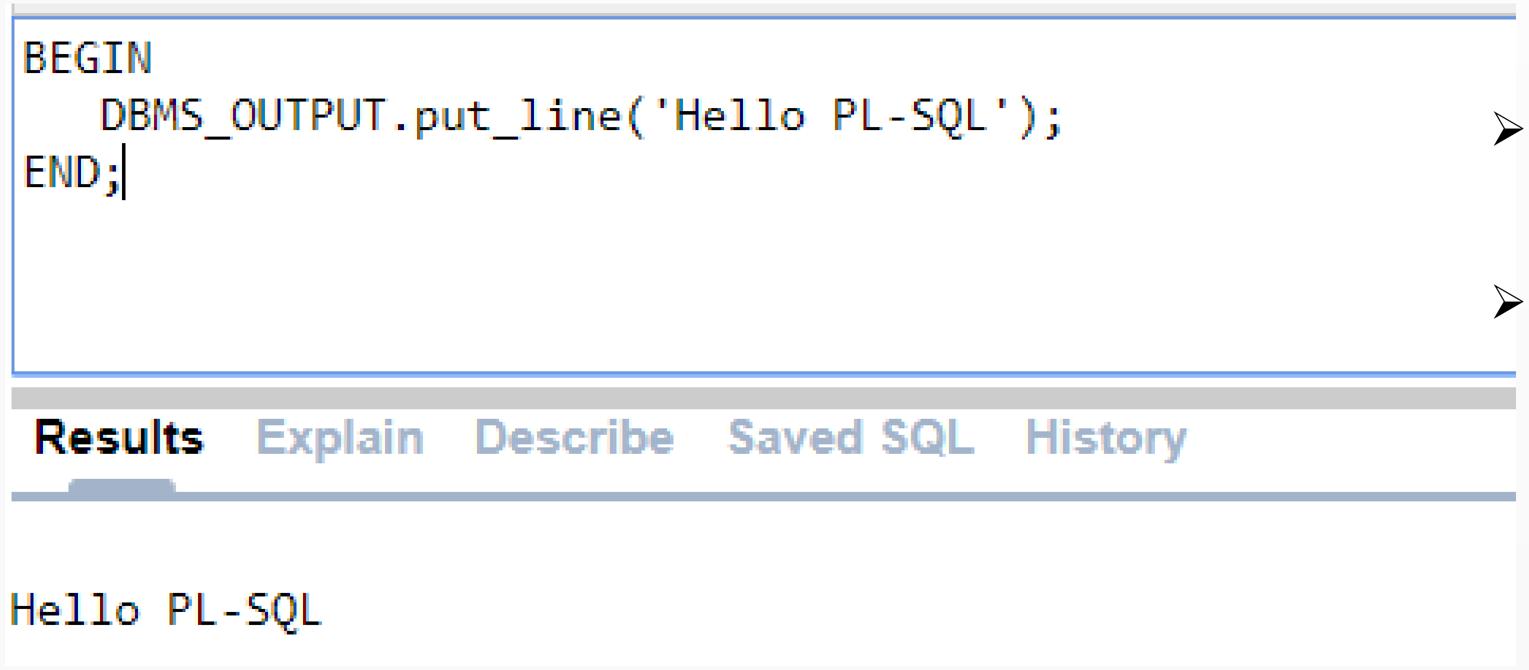




```
call mkj.getInstructorsCourses();
```

PL-SQL

- Pl – SQL is the blocked programming language for Database.
- Program units can be named or unnamed blocks.
- Unnamed blocks are known to be as anonymous blocks because its not going to be saved in the database. So it will never have name.
- We typically use such blocks whenever we need to perform one time activity.

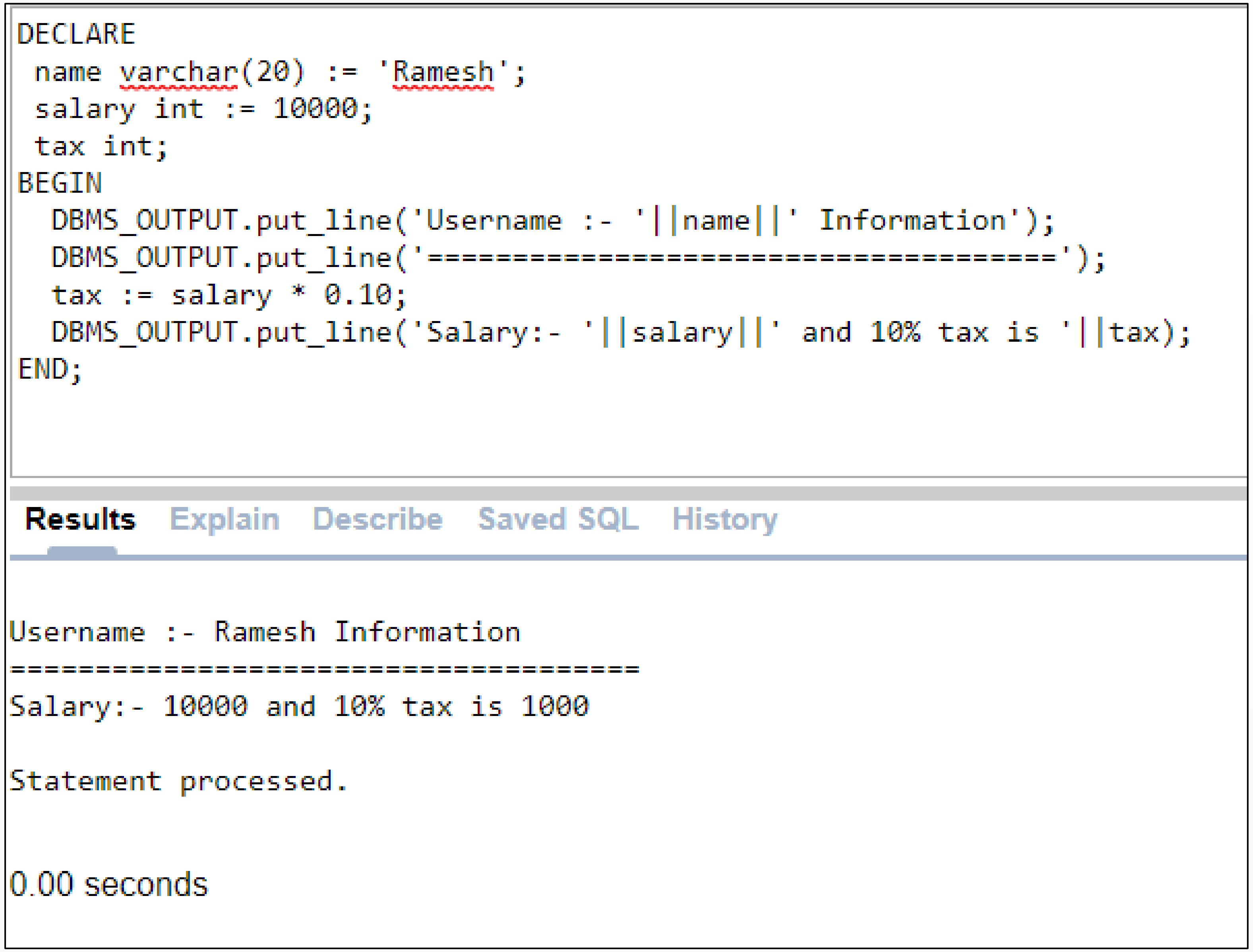


```
BEGIN
   DBMS_OUTPUT.put_line('Hello PL-SQL');
END;
```

```
Hello PL-SQL
```

- ➢ BEGIN and END are the starting and ending of the block.
- ➢ DBMS_OUTPUT is used to render output either on console or file system.

PL-SQL Continue..



```
DECLARE
 name varchar(20) := 'Ramesh';
 salary int := 10000;
 tax int;
BEGIN
  DBMS_OUTPUT.put_line('Username :- '||name||' Information');
  DBMS_OUTPUT.put_line('=====================================');
  tax := salary * 0.10;
  DBMS_OUTPUT.put_line('Salary:- '||salary||' and 10% tax is '||tax);
END;
```

```
Username :- Ramesh Information
=====================================
Salary:- 10000 and 10% tax is 1000

Statement processed.


0.00 seconds
```

- Declare section is used to declare and initialize variables.
  All variables consider as local scope.
- Assignment operator is

  :=

- Concatenation operator is **||**

3. PLSQL Script.txt

4. PLSQL Insert Script.txt

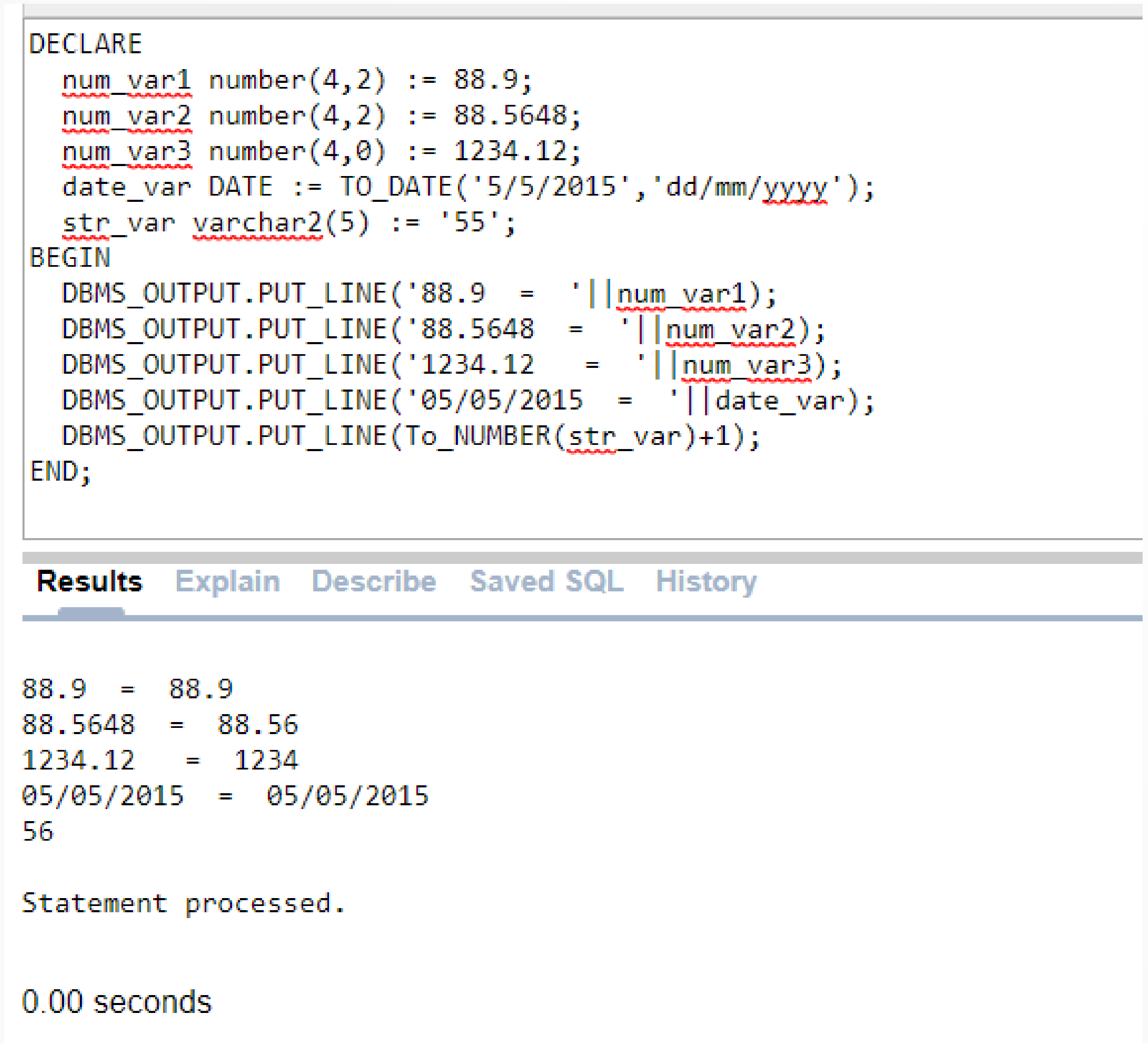
PL-SQL Data Types

Basic Data Types	
Varchar2	Variable length character , max size 4000 & min size should be 1.
Number(p,s)	P stands for precision and s stands for scale. The precision p can range from 1 to 38, and scale can have range from 84 to 127.
Long	Character data variable upto 2GB.
DATE	Valid date
Many more data type and sub data types are available , like Integer which is a sub type of Number data type, which takes only whole number.	

PL-SQL Type conversion

Type Conversion	
If we want to convert the values to the different data type , we should use type conversion functions.	
to_char(value,[format_mask])	• Use to convert number or date to String. • Format_mask is optional use to provide format to the String
To_number(String , [format_mask])	• Use to convert String to number and format_mask is optional , which is use to provide formatting
To_date(String,[format_mask])	• Use to convert String to date and format_mask is optional , which is use to provide formatted date.

PL-SQL



```
DECLARE
  num_var1 number(4,2) := 88.9;
  num_var2 number(4,2) := 88.5648;
  num_var3 number(4,0) := 1234.12;
  date_var DATE := TO_DATE('5/5/2015','dd/mm/yyyy');
  str_var varchar2(5) := '55';
BEGIN
  DBMS_OUTPUT.PUT_LINE('88.9   =  '||num_var1);
  DBMS_OUTPUT.PUT_LINE('88.5648   =  '||num_var2);
  DBMS_OUTPUT.PUT_LINE('1234.12    =  '||num_var3);
  DBMS_OUTPUT.PUT_LINE('05/05/2015   =  '||date_var);
  DBMS_OUTPUT.PUT_LINE(To_NUMBER(str_var)+1);
END;
```

Results   Explain   Describe   Saved SQL   History

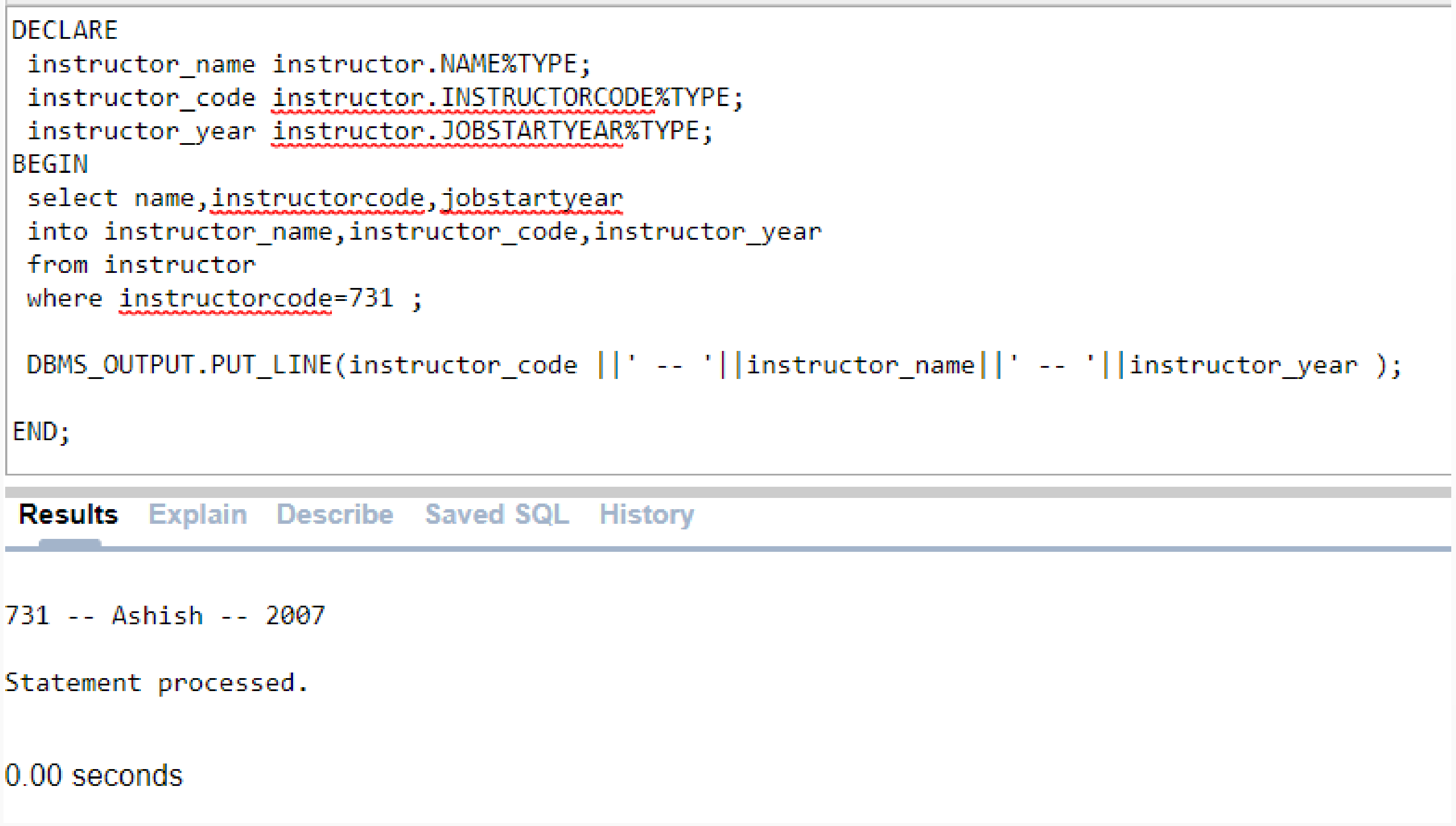
```
88.9   =  88.9
88.5648   =  88.56
1234.12    =  1234
05/05/2015   =  05/05/2015
56

Statement processed.

0.00 seconds
```

- Assigning any number more than 4 digits leads compile time error.
- Assigning date in different format leads compile time error.

5 PLSQL Formatting.txt

PL-SQL %TYPE & %ROWTYPE

- %TYPE used to assign datatype of a column .
- %ROWTYPE used to assigned entire row to the variable so that further value can be accessed easily.



```
DECLARE
 instructor_name instructor.NAME%TYPE;
 instructor_code instructor.INSTRUCTORCODE%TYPE;
 instructor_year instructor.JOBSTARTYEAR%TYPE;
BEGIN
 select name,instructorcode,jobstartyear
 into instructor_name,instructor_code,instructor_year
 from instructor
 where instructorcode=731 ;

 DBMS_OUTPUT.PUT_LINE(instructor_code ||' -- '||instructor_name||' -- '||instructor_year );

END;
```

```
731 -- Ashish -- 2007

Statement processed.
```

0.00 seconds

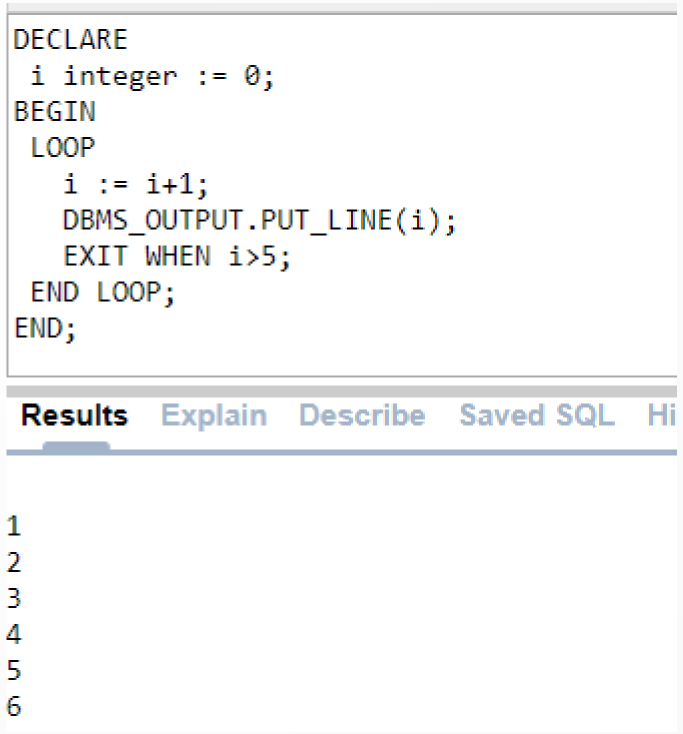
6. PER_TYPE Script.txt

7. PER_ROWTYPE Script.txt

PL-SQL Control Statements

```
IF <Condition>

THEN
          statements
          statements
ELSE
          statements
           statements

END IF;
```

8. IF ELSE SCRIPT.txt



```
DECLARE
 i integer := 0;
BEGIN
 LOOP
   i := i+1;
   DBMS_OUTPUT.PUT_LINE(i);
   EXIT WHEN i>5;
 END LOOP;
END;
```

```
1
2
3
4
5
6
```

Note:
The **PLSQL EXTRACT function** is used for extracting a specific value such as year, month, day or hour from a date

Syntax :

EXTRACT (field from source)

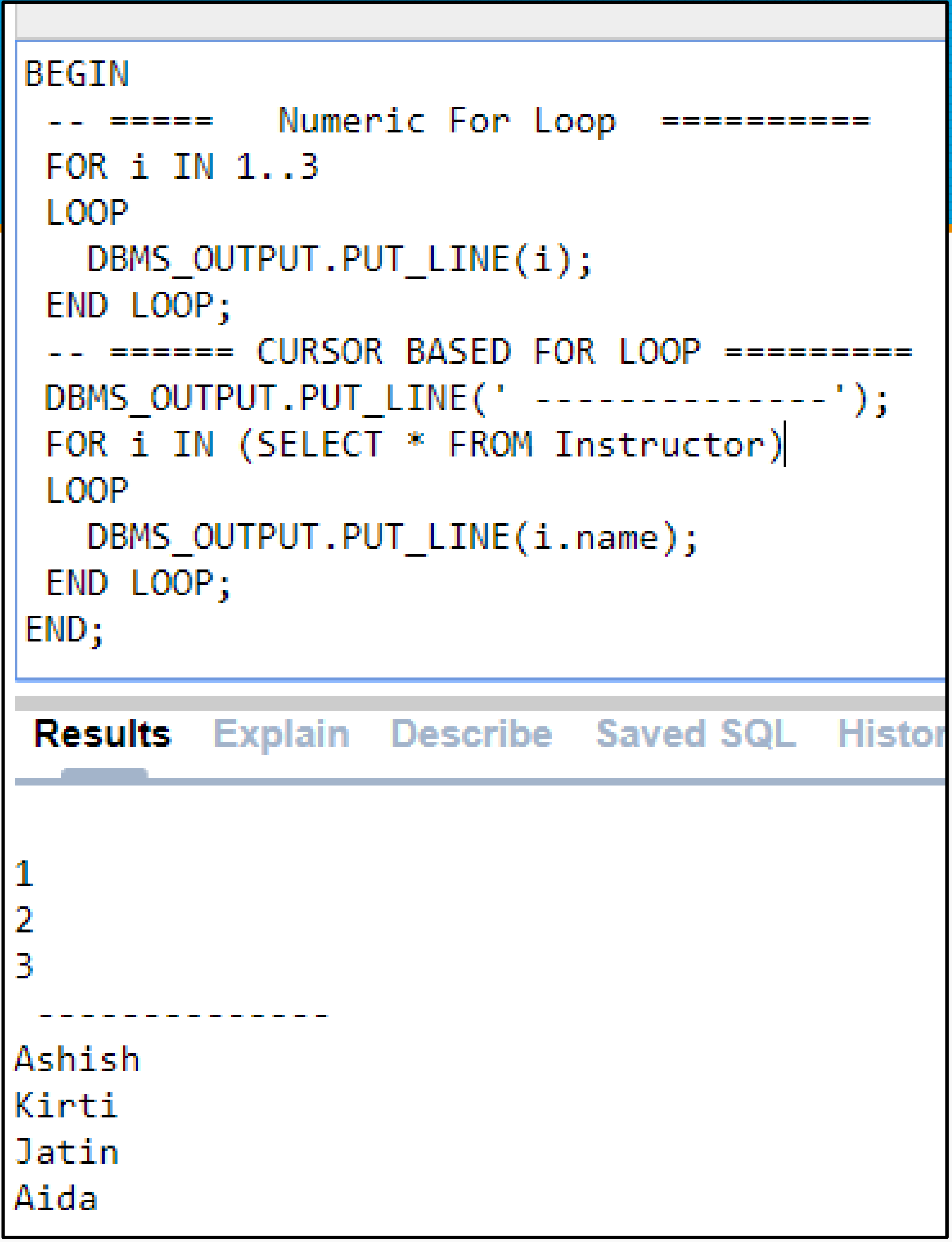
PL-SQL for loop

1. PL/SQL support two versions of for loop
   a) numeric and
   b) cursor FOR loop
2. The numeric for loop iterates across defined range.
3. While the cursor for loop iterates over returned select statements.

Syntax:

```
For i IN x..y
LOOP
   statements
END LOOP;
```

```
For i IN (select statement)
LOOP
   statements
END LOOP;
```



```
BEGIN
 -- =====   Numeric For Loop   ==========
 FOR i IN 1..3
 LOOP
   DBMS_OUTPUT.PUT_LINE(i);
 END LOOP;
 -- ====== CURSOR BASED FOR LOOP =========
 DBMS_OUTPUT.PUT_LINE(' --------------');
 FOR i IN (SELECT * FROM Instructor)
 LOOP
   DBMS_OUTPUT.PUT_LINE(i.name);
 END LOOP;
END;
```

```
1
2
3
 --------------
Ashish
Kirti
Jatin
Aida
```

PL-SQL - Procedures

1. **Stored Procedures** are created to perform one or more DML operations on Database.
2. It is nothing but the group of SQL statements that accepts some input in the form of parameters and performs some task and may or may not returns a value.
3. Both function as well as stored procedure have a **unique named block** of code which is compiled and stored in the database.
4. The most important part is parameters. Parameters are used to pass values to the Procedure. There are 3 different types of parameters, they are as follows:
    - IN
    - OUT
    - IN OUT

IN	**IN** mode refers to **READ ONLY mode** which is used for a variable by which it will accept the value from the user. It is the default parameter mode.
OUT	**OUT** mode refers to **WRITE ONLY mode** which is used for a variable that will return the value to the user.
IN OUT	**IN OUT** mode refers to **READ AND WRITE mode** which is used for a variable that will either accept a value from the user or it will return the value to the user.

MKJ IT Learnings

Training for professionals

Training For Professionals

Enhance Your Technical Skills and Become Future Ready