

# Topics

---

1. PL/SQL review
2. Cursors
3. Functions
4. MongoDB

# PL/SQL Review

---

- What are the different types of iteration statements?
- Basic LOOP
- FOR LOOP
- Cursor FOR LOOP
- WHILE LOOP
- How can you leave a loop
  - EXIT
  - EXIT WHEN
- What does Continue or Continue When do?

# Select Into

- The Select Into statement used in a procedure's executable block can have three outcomes
- It retrieves a row
- No Data Found
- Too Many Rows
- If more than one row can be returned
- Use a Cursor

# CURSORS

---

- Cursors are used to process multiple rows in PL/SQL blocks.
- In this course, we learn basic fundamentals about cursors. We use cursors to return multiple rows from a PL/SQL procedure to a caller procedure or program.
- A cursor is a pointer to a context area that includes the result of a processed SQL statement.
- Simply, a cursor contains the rows of a select statement.
- In PL/SQL, cursors are used to access and process the rows returned by a SELECT statement.
- There are two types of cursors:
  - Implicit cursors
  - Explicit cursors

# Implicit Cursors

- The implicit cursors are automatically created while a statement such as INSERT, UPDATE, DELETE, and SELECT are executed.
- The implicit cursor attributes can be used to determine if any rows have been affected as a result of the execution of a SQL statement.
- In PL/SQL, the SELECT INTO statement, the implicit cursor can be evaluated to see if any row is returned.

# Implicit Cursors

Attributes	Description
SQL%FOUND	It returns true if at least one row is affected by the execution of a DML or a SELECT statement
SQL%NOTFOUND	It returns true if no row is affected by the execution of a DML or a SELECT statement
SQL%ROWCOUNT	It returns the number of rows affected by the execution of a DML statement or a SELECT statement
SQL%ISOPEN	SQL%ISOPEN always has the value FALSE.

- Exceptions with SELECT INTO
- NO DATA FOUND
- TOO MANY ROWS RETURNED
- If multiple rows are returned we replace SELECT INTO with a SELECT within a CURSOR
- Explicit Cursor

# Explicit Cursors

- The explicit cursors are defined in the declaration section of a PL/SQL block by programmers. It is used to process the multi-row results from a SELECT statement.
- To define cursor:
  - **CURSOR cursor\_name IS select\_statement;**
- To use an explicit cursor in a PL/SQL block:
  - Declare the cursor
  - The memory is allocated for the cursor
  - Open the cursor
  - Retrieve data from the cursor
  - Close the cursor
  - The memory allocated to the cursor is released.

# Explicit Cursors - DECLARE

- Cursors can be defined in the declaration section.
- **CURSOR cursor\_name IS select\_statement;**
- **DECLARE**
- **CURSOR cursor\_1 IS**
  - **SELECT last\_name, job\_title FROM employees**
  - **WHERE job\_title LIKE 'VP%**'
  - **ORDER BY last\_name;**
- After the cursor is opened and a temporary table is loaded, you want to fetch the row column data a row at a time and put it into your appropriate variables.
- You can use **%type** to pick up the same definition for your variable that is used in the original table column

# Cursor Defined

- You specify local variables and define the cursor in the declaration section.
- DECLARE
  - e\_last\_name employees.lastname%type;
  - e\_job\_title employees.jobtitle%type;
- CURSOR emp\_cursor IS
  - SELECT lastname, jobtitle
  - FROM employees
  - WHERE jobtitle LIKE 'VP%'
  - ORDER BY lastname;

# Open a Cursor

---

- You open and use the cursor in the executable section.
- BEGIN
- `OPEN emp_cursor;`

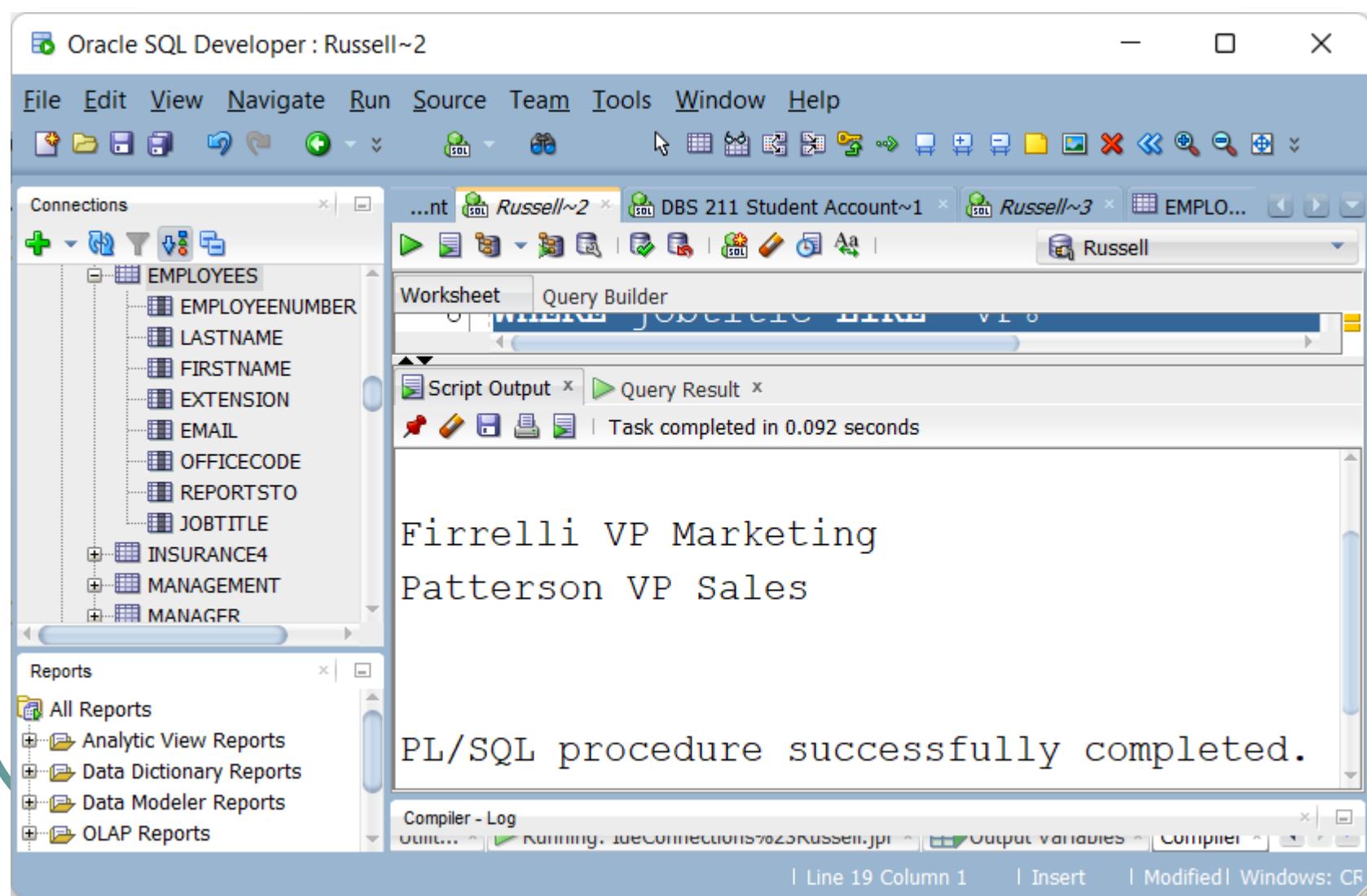
# Fetching Data using a Cursor

- In the executable section, you can fetch data from a cursor row by row in a loop.
- The syntax is:
  - `FETCH cursor_name INTO variable_list;`
  - `BEGIN`
  - `OPEN emp_cursor;`
  - `LOOP`
  - `FETCH emp_cursor into e_last_name, e_job_title;`
  - `EXIT WHEN emp_cursor%notfound;`
  - `dbms_output.put_line(e_last_name || ' ' || e_job_title);`
  - `END LOOP;`

# Close a Cursor

- You close a cursor in the executable section.
- BEGIN
- OPEN emp\_cursor;
- LOOP
- FETCH emp\_cursor into e\_last\_name, e\_job\_title;
- EXIT WHEN emp\_cursor%notfound;
- dbms\_output.put\_line(e\_last\_name || ' ' || e\_job\_title);
- END LOOP;
- CLOSE emp\_cursor;
- END;

# Executing the Cursor Code



# Explicit Cursor with Parameters

- You can pass parameters that affect what rows are returned when the cursor is opened.
- If you want to return all products with prices between two prices, you can pass an upper and lower price
- The following anonymous block has hard coded 115 and 120 as the upper and lower price to be used by the cursor

# %Type

- When you use **%type**, you do not have to know the definition of a column in a table when setting up your variable.
- Also, if the type definition for that column changes, your code does not have to be updated – it will automatically pick up the new type definition

# Explicit Cursor with Parameters

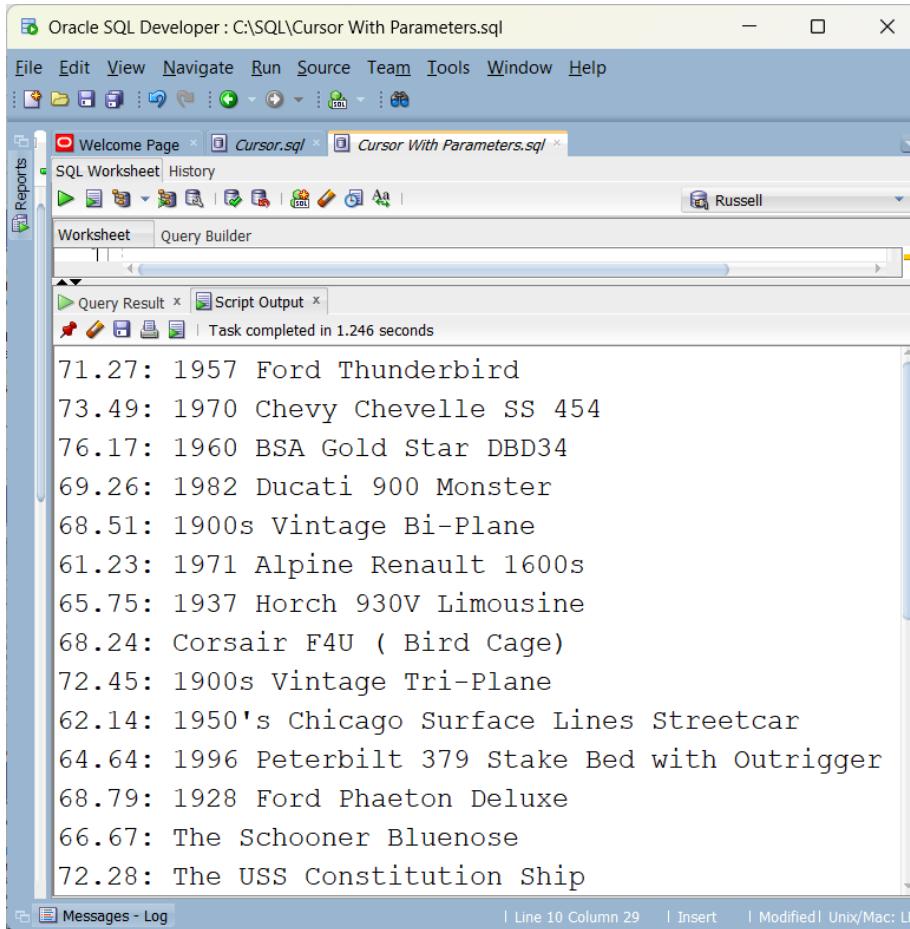
- Can a cursor be flexible, by accepting different parameters each time it is used?
- We have a PRODUCTS table with MSRP amounts.
- One time we can have a temporary result table with prices between 115 and 120
- Another time that cursor will produce a temporary result table with prices between 60 and 80

# Explicit Cursor with Parameters

The screenshot shows the Oracle SQL Developer interface. The title bar reads "Oracle SQL Developer : Russell~2". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar has various icons for connection management, code navigation, and database operations. The Connections sidebar lists several schemas: ORDERS, PART, PATIENT4, PAYMENTS, PRODUCTLINES, and PRODUCTS. The PRODUCTS schema is expanded, showing columns: PRODUCTCODE, PRODUCTNAME, PRODUCTLINE, PRODUCTSCALE, PRODUCTVENDOR, PRODUCTDESCRIPTION, QUANTITYINSTOCK, BUYPRICE, and MSRP. The Reports sidebar lists All Reports, Analytic View Reports, Data Dictionary Reports, Data Modeler Reports, OLAP Reports, TimesTen Reports, and User Defined Reports. The central workspace is titled "Worksheet" and contains a "Script Output" tab showing the results of a query. The results are as follows:

```
117.44: 1952 Citroen-15CV
118.28: 1992 Porsche Cayenne Turbo Silver
115.75: Diamond T620 Semi-Skirted Tanker
118.65: ATA: B757-300
118.94: 1996 Moto Guzzi 1100i
117.44: 1968 Dodge Charger
118.5: 1957 Chevy Pickup
115.16: 1969 Dodge Charger
116.67: 1940 Ford Pickup Truck
```

# Explicit Cursor with Parameters



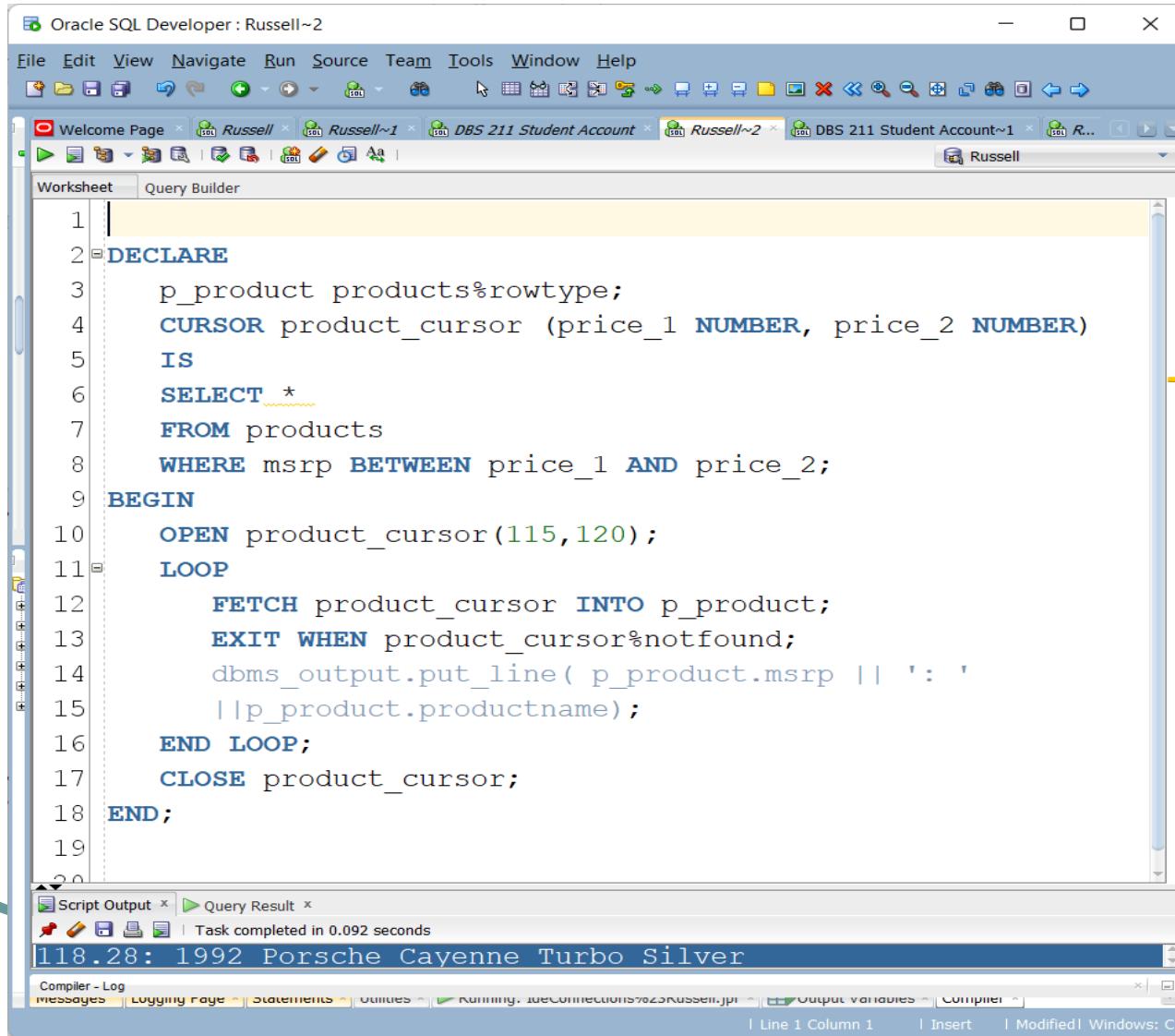
The screenshot shows the Oracle SQL Developer interface with a query window titled "Cursor With Parameters.sql". The query results display a list of items, each consisting of a numerical value followed by a colon, a year, a manufacturer, and a model name. The results are as follows:

```
71.27: 1957 Ford Thunderbird
73.49: 1970 Chevy Chevelle SS 454
76.17: 1960 BSA Gold Star DBD34
69.26: 1982 Ducati 900 Monster
68.51: 1900s Vintage Bi-Plane
61.23: 1971 Alpine Renault 1600s
65.75: 1937 Horch 930V Limousine
68.24: Corsair F4U ( Bird Cage)
72.45: 1900s Vintage Tri-Plane
62.14: 1950's Chicago Surface Lines Streetcar
64.64: 1996 Peterbilt 379 Stake Bed with Outrigger
68.79: 1928 Ford Phaeton Deluxe
66.67: The Schooner Bluenose
72.28: The USS Constitution Ship
```

# %Rowtype

- Used to pull in the column type definitions for an entire row of a table
- `variable_name table_or_view_name%ROWTYPE;`
- For every column of the table or view, the record has a field with the same name and data type.
- PRODUCTS has ProductName, MSRP
- `p_product PRODUCTS%ROWTYPE`
- We now have variables called `p_product.productname` and `p_product.msrp`
- Which can be loaded from the table's ProductName and MSRP columns and the column types match

# Explicit Cursor with Parameters

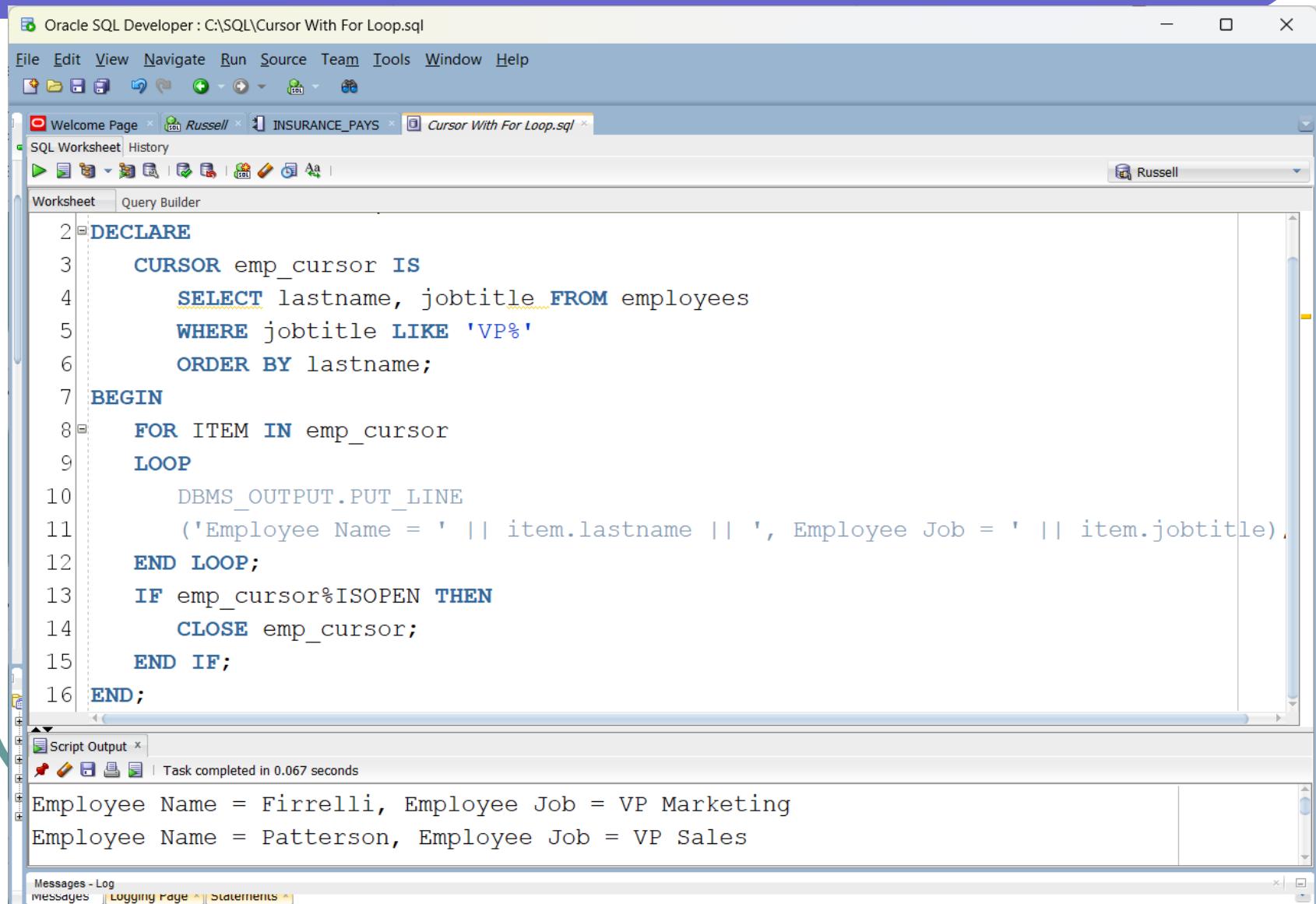


The screenshot shows the Oracle SQL Developer interface with the following details:

- Title Bar:** Oracle SQL Developer : Russell~2
- Menu Bar:** File, Edit, View, Navigate, Run, Source, Team, Tools, Window, Help
- Toolbar:** Standard toolbar with various icons.
- Tab Bar:** Welcome Page, Russell, Russell~1, DBS 211 Student Account, Russell~2, DBS 211 Student Account~1, R... (active tab)
- Worksheet Tab:** Russell
- Code Area:** Displays the following PL/SQL code:

```
1
2  DECLARE
3      p_product products%rowtype;
4      CURSOR product_cursor (price_1 NUMBER, price_2 NUMBER)
5  IS
6      SELECT *
7      FROM products
8      WHERE msrp BETWEEN price_1 AND price_2;
9  BEGIN
10     OPEN product_cursor(115,120);
11    LOOP
12        FETCH product_cursor INTO p_product;
13        EXIT WHEN product_cursor%notfound;
14        dbms_output.put_line( p_product.msrp || ':' ||
15                               p_product.productname);
16    END LOOP;
17    CLOSE product_cursor;
18 END;
```
- Script Output Tab:** Shows the output of the executed query: 118.28: 1992 Porsche Cayenne Turbo Silver
- Compiler - Log Tab:** Shows the log message: Task completed in 0.092 seconds
- Status Bar:** Line 1 Column 1, Insert, Modified, Windows: CR

# Explicit Cursor in FOR LOOP



The screenshot shows the Oracle SQL Developer interface with the following details:

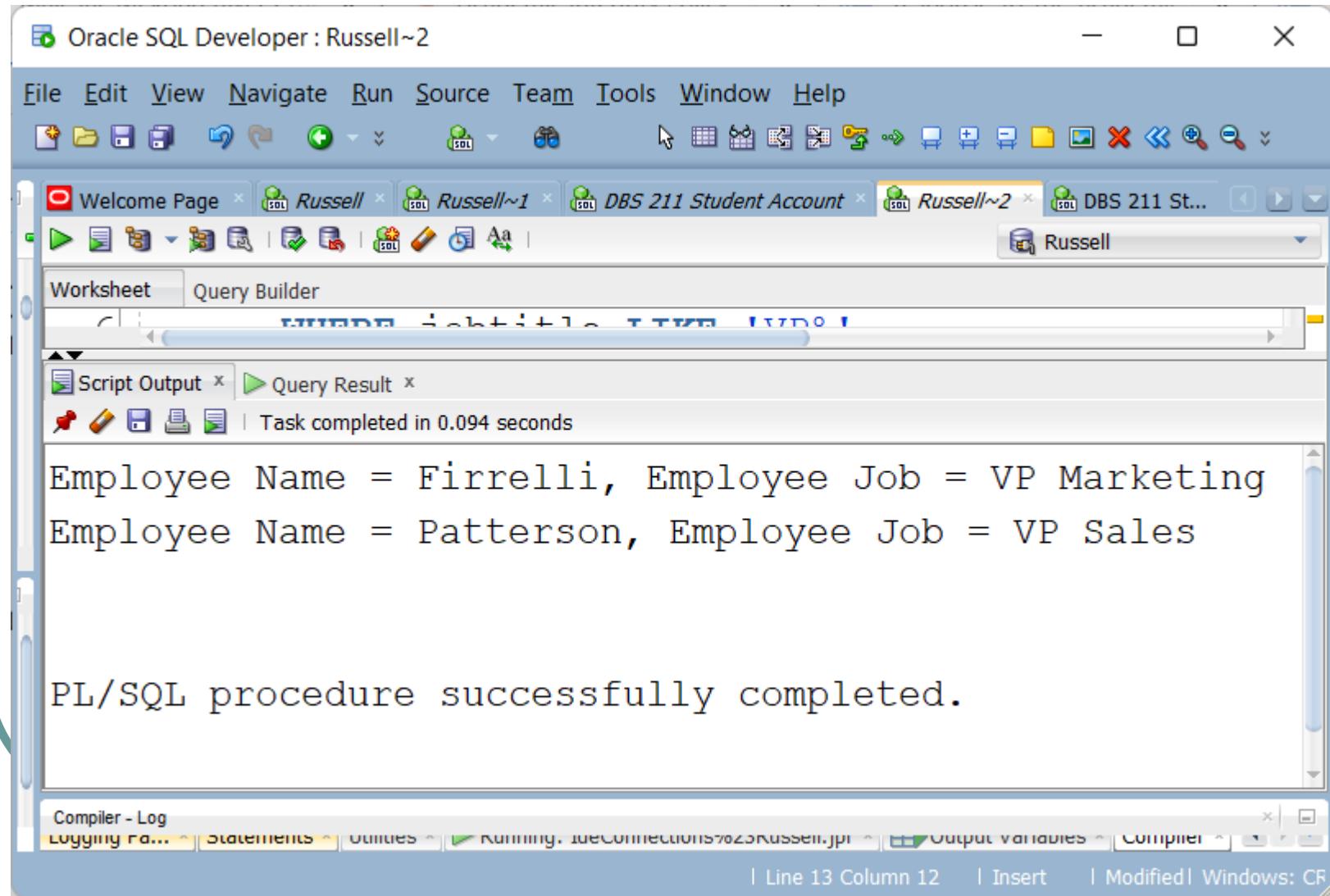
- Title Bar:** Oracle SQL Developer : C:\SQL\Cursor With For Loop.sql
- Menu Bar:** File Edit View Navigate Run Source Team Tools Window Help
- Toolbar:** Standard toolbar with icons for New, Open, Save, Print, etc.
- Tab Bar:** Welcome Page, Russell, INSURANCE\_PAY\$ (selected), Cursor With For Loop.sql
- Worksheet Tab:** SQL Worksheet (selected) and History
- Worksheet Content:** PL/SQL code listing rows 2 through 16. The code declares a cursor for employees with job titles like 'VP%' and prints them to the screen.

```
2 DECLARE
3   CURSOR emp_cursor IS
4     SELECT lastname, jobtitle FROM employees
5     WHERE jobtitle LIKE 'VP%'
6     ORDER BY lastname;
7 BEGIN
8   FOR ITEM IN emp_cursor
9   LOOP
10    DBMS_OUTPUT.PUT_LINE
11      ('Employee Name = ' || item.lastname || ', Employee Job = ' || item.jobtitle),
12  END LOOP;
13  IF emp_cursor%ISOPEN THEN
14    CLOSE emp_cursor;
15  END IF;
16 END;
```

- Script Output Window:** Shows the output of the executed code:

```
Employee Name = Firrelli, Employee Job = VP Marketing
Employee Name = Patterson, Employee Job = VP Sales
```
- Bottom Navigation:** Messages - Log, Messages, Logging Page, Statements

# Explicit Cursor in FOR LOOP



The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : Russell~2". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar contains various icons for database management tasks. The connection bar at the top has several entries, with "Russell~2" currently selected. The main workspace shows a "Worksheet" tab active, displaying the following PL/SQL code and its output:

```
Employee Name = Firrelli, Employee Job = VP Marketing
Employee Name = Patterson, Employee Job = VP Sales

PL/SQL procedure successfully completed.
```

The bottom status bar indicates "Compiler - Log" and "Running: 108CONNECTIONS7025RUSSELL.jpr". The status bar also shows the current line and column (Line 13 Column 12), and options for Insert, Modified, and Windows.

# Explicit Cursor Attributes

Attributes	Description
%FOUND	INVALID_CURSOR: if the cursor is not open NULL: before we fetch the first row FALSE: if no row is fetched successfully TRUE: if the row is fetched successfully
%NOTFOUND	INVALID_CURSOR: if the cursor is not open. NULL: before we fetch the first row
%ROWCOUNT	INVALID_CURSOR: if the cursor is not open Otherwise: It returns the number of rows returned from the cursor
%ISOPEN	TRUE: if the cursor is open FALSE: if the cursor is not open

# %RowCount

- CursorName%RowCount
- Cursors do not know how many rows there are when the cursor is opened
- The count is incremented with each successful Fetch
- You can use this feature within a loop while fetching rows or after the loop is finished to report on total rows fetched

# %RowCount

Oracle SQL Developer : C:\SQL\Cursors\Cursor With RowCount.sql

File Edit View Navigate Run Source Team Tools Window Help

Welcome Page UPDATE\_STAFF1.sql DBS311 Fall 24 ZB~22 Cursor With RowCount.sql

SQL Worksheet History 0.087 seconds DBS311 Fall 24 ZB

Worksheet Query Builder

```
10      OPEN emp_cursor;
11      DBMS_OUTPUT.Put_Line ('Cursor opened and row count is ' || emp_cursor%RowCount );
12  LOOP
13      FETCH emp_cursor into e_last_name, e_job_title;
14      EXIT WHEN emp_cursor%notfound;
15      dbms_output.put_line(e_last_name || ' ' || e_job_title);
16  END LOOP;
17      DBMS_OUTPUT.Put_Line (emp_cursor%RowCount || ' employees found' );
18      CLOSE emp_cursor;
19  END;
```

Script Output Task completed in 0.087 seconds

Cursor opened and row count is 0  
Firrelli VP Marketing  
Patterson VP Sales  
2 employees found

Statements - Log Messages Logging Page Statements

Saved: DBS311 Fall 24 ZB~21 Line 1 Column 1 Insert Unix/Mac: LF

# Cursor CountriesCursor is:

Oracle SQL Developer : A DBS311 Student Account F25~2

File Edit View Navigate Run Source Team Tools Window Help

Worksheet Query Builder

```
1 select country, count(*) TOTALCUSTOMERS
2 from customers
3 group by country
4 having count(*) > 7
5 order by country;
```

Query Result x

All Rows Fetched: 3 in 1.858 seconds

COUNTRY	TOTALCUSTOMERS
France	12
Germany	13
USA	36

Messages - Log

Statements

Line 1 Column 40 | Insert | Modified | Windows: CR

Oracle SQL Developer : A DBS311 Student Account F25~2

File Edit View Navigate Run Source Team Tools Window Help

Worksheet Query Builder

```
1 select country, count(*) TOTALCUSTOMERS
2 from customers
3 group by country
4 having count(*) > NUMIN
5 order by country;
```

Query Result x

All Rows Fetched: 3 in 1.858 seconds

COUNTRY	TOTALCUSTOMERS
France	12
Germany	13
USA	36

Messages - Log

Statements

Line 4 Column 26 | Insert | Modified | Windows: CR

Oracle SQL Developer : C:\SQL\DBS311\Assignment & Labs F25\LAB7 TEST TOTALCOUNTRY.sql

File Edit View Navigate Run Source Team Tools Window Help



Welcome Page A DBS311 Student Account F25~2 CUSTCOUNTRY LAB7 TEST TOTALCOUNTRY.sql

SQL Worksheet History



Worksheet Query Builder

```
1 set serveroutput on;
2 DECLARE
3     TOTALC      NUMBER;
4     NOCUSTOMERS NUMBER;
5 BEGIN
6     NOCUSTOMERS := 7;
7     CUSTCOUNTRY(NOCUSTOMERS, TOTALC);
8     DBMS_OUTPUT.PUT_LINE ('There are ' || TotalC || ' countries with more than ' ||
9                           NOCUSTOMERS || ' customers');
10 END;
```

Messages - Log  
messages Logging Page Statements

| Line 1 Column 1 | Insert | Unix/Mac: LF

# USER DEFINED FUNCTIONS

---

- Created with PL/SQL or Java WITH ORACLE
- Created with SQL/PL or RPGLE, C ... programs with DB2
- UDF's can appear in an SQL statement
- A UDF in a SELECT statement can be found:
  - in the select list
  - as the condition in a WHERE clause
  - in an ORDER BY clause
  - in a GROUP BY clause

# USER DEFINED FUNCTIONS

---

- UDF's can appear in other SQL statements
- A UDF in an INSERT statement can be found:
  - in the VALUES clause
- A UDF in an UPDATE statement can be found:
  - in the WHERE clause
- A PL/SQL function is a stored piece of code that can be called from other functions/procedures or programs. A function returns a value using the RETURN clause.

# UDF with a passed parameter

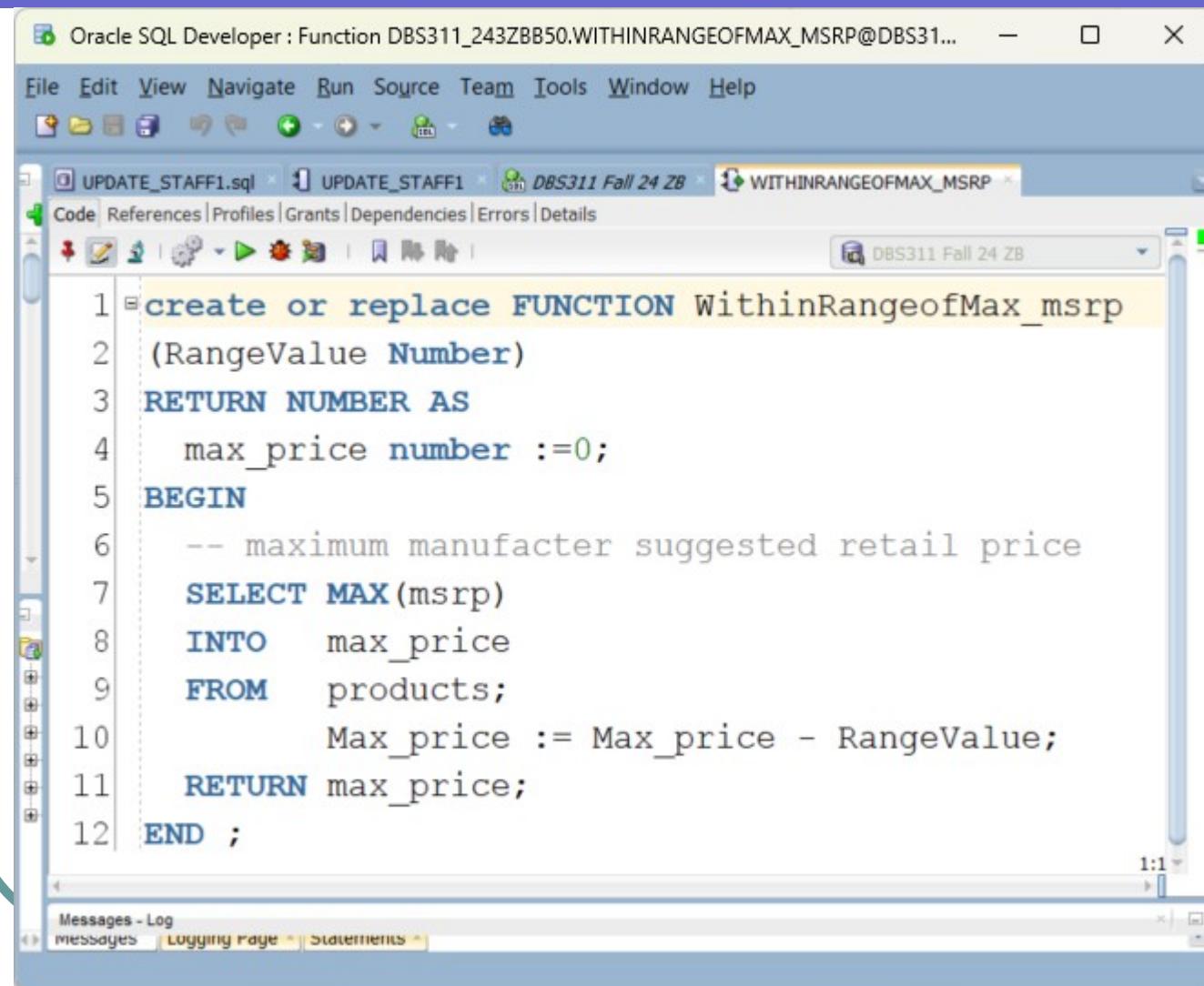
The screenshot shows the Oracle SQL Developer interface with a query window displaying a SELECT statement. The statement retrieves product names and MSRP from the PRODUCTS table where the MSRP is greater than or equal to a value obtained from a user-defined function (UDF) named WITHINRANGEOFMAX\_MSRP, which takes a parameter of 20.

```
1 | SELECT PRODUCTNAME, MSRP
2 | FROM PRODUCTS
3 | WHERE MSRP>= WITHINRANGEOFMAX_MSRP(20);
```

The query result shows three rows of data:

	PRODUCTNAME	MSRP
1	1952 Alpine Renault 1300	214.3
2	1968 Ford Mustang	194.57
3	2001 Ferrari Enzo	207.8

# UDF with a passed parameter



The screenshot shows the Oracle SQL Developer interface with a blue header bar. The main window displays a SQL script for creating a function named `WithinRangeofMax_msrp`. The script uses a passed parameter `RangeValue` and returns a number. It selects the maximum manufacturer suggested retail price from the `products` table and then subtracts the `RangeValue` from it. The code is highlighted in yellow.

```
1 create or replace FUNCTION WithinRangeofMax_msrp
2 (RangeValue Number)
3 RETURN NUMBER AS
4     max_price number :=0;
5 BEGIN
6     -- maximum manufacter suggested retail price
7     SELECT MAX(msrp)
8     INTO   max_price
9     FROM   products;
10    Max_price := Max_price - RangeValue;
11    RETURN max_price;
12 END ;
```

The interface includes a toolbar, a tabs section with `UPDATE_STAFF1.sql`, `UPDATE_STAFF1`, `DBS311 Fall 24 ZB`, and `WITHINRANGEOMAX_MSRP`, and a bottom pane titled "Messages - Log" with tabs for "Messages", "Logging Page", and "Statements".

# A UDF with two input parameters

The screenshot shows the Oracle SQL Developer interface. The title bar reads "Oracle SQL Developer : DBS311 Fall 24 ZB~15". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar has various icons for file operations and SQL. The tabs at the top show "Welcome Page", "Russell", "WITHIN50OFMAX\_MSRP", "DBS311 Fall 24 ZB~15" (which is the active tab), and "WITHINRANGEOFMAX\_MSRP.sql". The bottom tabs show "Script Output" and "Query Result". The "Query Result" tab is selected and displays the output of the following query:

```
9  SELECT PRODUCTNAME,
10    BUYPRICE,
11    MSRP,
12    PRICEMARKUP(MSRP, BUYPRICE) MARKUP
13   FROM PRODUCTS;
```

The output table has columns: PRODUCTNAME, BUYPRICE, MSRP, and MARKUP. The data is as follows:

PRODUCTNAME	BUYPRICE	MSRP	MARKUP
Boeing X-32A JSF	32.77	49.66	16.89
Pont Yacht	33.3	50.31	17.01
1969 Harley Davidson Ultimate Chopper	48.81	95.7	46.89
1952 Alpine Renault 1300	98.58	214.3	115.72
1996 Moto Guzzi 1100i	68.99	118.94	49.95
2003 Harley-Davidson Eagle Drag Bike	91.02	193.66	102.64
1972 Alfa Romeo GTA	85.68	136	50.32
1962 Lancia Delta 16V	103.42	147.74	44.32
1968 Ford Mustang	95.34	194.57	99.23
2001 Ferrari Enzo	95.59	195.33	99.74
1958 Setra Bus	77.9	136.67	58.77
2002 Suzuki XREO	66.27	150.62	84.35
1969 Corvair Monza	89.14	151.08	61.94
1968 Dodge Charger	75.16	117.44	42.28

# A UDF with two input parameters

---

- create or replace FUNCTION PRICEMARKUP
- ( MSRP IN FLOAT,
- BUYPRICE IN FLOAT
- )
- RETURN FLOAT AS
- MARKUP FLOAT;
- BEGIN
- MARKUP := MSRP - BUYPRICE;
- RETURN(MARKUP);
- END;

er By

P

SC;

Oracle SQL Developer : Russell

File Edit View Navigate Run Source Team Tools Window Help

Welcome Page Russell PRICEMARKUP

Worksheet Query Builder

```
1 SELECT PRODUCTNAME,
2        BUYPRICE,
3        MSRP,
4        PRICEMARKUP(MSRP, BUYPRICE) MARKUP
5 FROM PRODUCTS
6 WHERE PRICEMARKUP(MSRP, BUYPRICE) > 60
7 ORDER BY PRICEMARKUP(MSRP, BUYPRICE) DESC;
```

Script Output x Query Result x

SQL | All Rows Fetched: 20 in 2.238 seconds

	PRODUCTNAME	BUYPRICE	MSRP	MARKUP
1	1952 Alpine Renault 1300	98.58	214.3	115.72
2	2001 Ferrari Enzo	95.59	207.8	112.21
3	2003 Harley-Davidson Eagle Drag Bike	91.02	193.66	102.64
4	1968 Ford Mustang	95.34	194.57	99.23
5	1928 Mercedes-Benz SSK	72.56	168.75	96.19
6	1992 Ferrari 360 Spider red	77.9	169.34	91.44
7	1969 Ford Falcon	83.05	173.02	89.97
8	2002 Suzuki XREO	66.27	150.62	84.35
9	1917 Grand Touring Sedan	86.7	170	83.3
10	1980s Black Hawk Helicopter	77.27	157.69	80.42
11	1948 Porsche Type 356 Roadster	62.16	141.28	79.12
12	1957 Corvette Convertible	69.93	148.8	78.87
13	1999 Indy 500 Monte Carlo SS	56.76	132	75.24
14	1976 Ford Gran Torino	73.49	146.99	73.5
15	1932 Model A Ford J-Coupe	58.48	127.13	68.65
16	1903 Ford Model A	68.3	136.59	68.29
17	1962 Volkswagen Microbus	61.34	127.79	66.45
18	1957 Chevv Pickup	55.7	118.5	62.8
19	1998 Chrysler Plymouth Prowler	101.51	163.73	62.22
20	1969 Corvair Monza	89.14	151.08	61.94

Messages - Log

Messages Loading Page Statements

Oracle SQL Developer : Russell

File Edit View Navigate Run Source Team Tools Window Help

Welcome Page Russell PRICEMARKUP

Russell

Worksheet Query Builder

```
1 create table productbk as
2 (select * from products);
3
4 UPDATE PRODUCTBK
5 SET MSRP = MSRP * .9
6 WHERE PRICEMARKUP(MSRP, BUYPRICE) > 60;
```

Script Output Query Result

Task completed in 2.535 seconds

Table PRODUCTBK created.

20 rows updated.

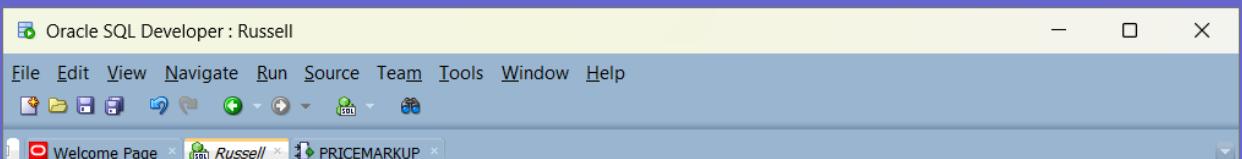
Messages - Log

Messages Logging Page Statements

IdeConnections%23Russell PRICEMARKUP | Line 6 Column 35 | Insert | Modified | Windows: CR

ENT

kup. We can  
markup is

A screenshot of the Oracle SQL Developer interface showing a query editor window titled "Oracle SQL Developer : Russell". The menu bar and toolbar are identical to the main window. The tab bar shows "Welcome Page", "Russell", and "PRICEMARKUP". The main area displays the following SQL query:

```
1
2 SELECT PRODUCTNAME, BUYPRICE, MSRP,
3         PRICEMARKUP(MSRP, BUYPRICE) MARKUP
4 FROM PRODUCTBK
5 WHERE PRICEMARKUP(MSRP, BUYPRICE) > 60
6 ORDER BY PRICEMARKUP(MSRP, BUYPRICE) DESC;
```

A screenshot of the Oracle SQL Developer interface showing a query result table titled "Query Result". The table displays 13 rows of data from the "PRODUCTBK" table, ordered by PRICEMARKUP(MSRP, BUYPRICE) in descending order. The columns are PRODUCTNAME, BUYPRICE, MSRP, and MARKUP. The data includes various high-end vehicles and their purchase and sale prices along with their respective markups.

# Relational Database Alternatives

---

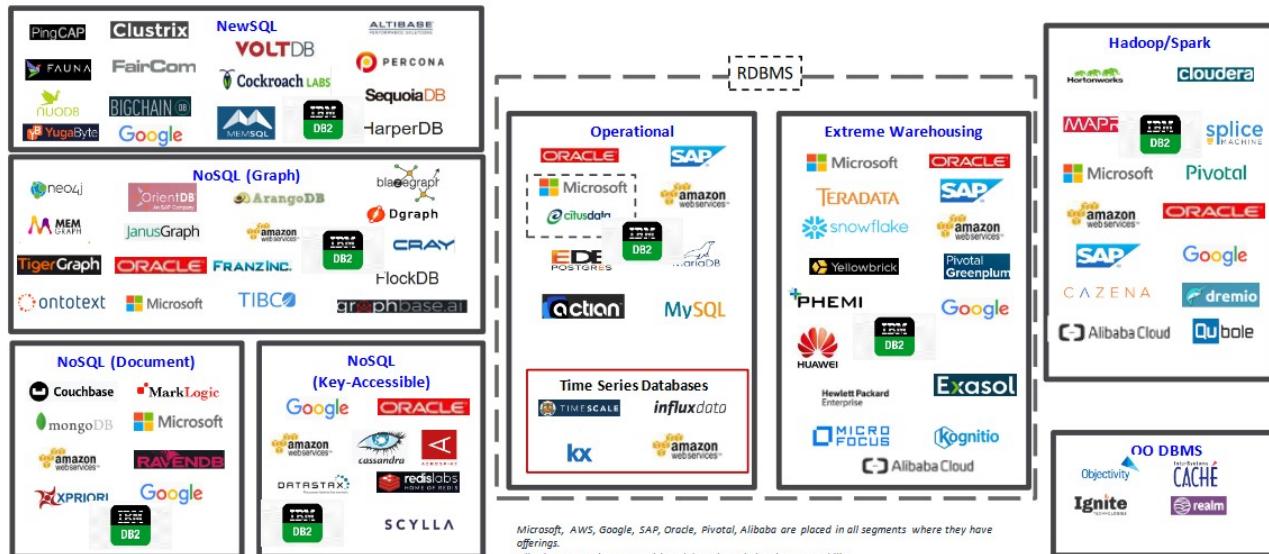
- NoSQL
- NewSQL
- Multi Model

# Wikipedia Definitions

---

- NoSQL – “Not only SQL” – A NoSQL database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases.
- NewSQL - is a class of relational database management systems that seek to provide the scalability of NoSQL systems for online transaction processing workloads while maintaining the ACID guarantees of a traditional database system
- Multi-model - Most database management systems are organized around a single data model that determines how data can be organized, stored, and manipulated. In contrast, a multi-model database is designed to support multiple data models against a single, integrated backend

# Database Landscape



# NoSQL

- A class of database management systems that depart from traditional RDBMSs
- Does not use SQL as the primary query language
- Is “schema-less”
  - No rigid schema enforced by the DBMS
  - Programmer-friendly for adding fields to a document
  - Might not guarantee full ACID behavior
  - Often has a distributed, fault-tolerant, elastic architecture
  - Highly optimized for retrieve and append operations over great quantities of data

# ACID Transactions

---

- **Atomicity** - all operations in a transaction succeed or are all rolled back
- **Consistency** - after transactions are complete, the database remains consistent
- **Isolation** - transactions are executed without interfering with other transactions
- **Durability** - after transactions are completed, changes will be permanent

# MongoDB

- What is MongoDB?
- MongoDB is a document-oriented database and differs from a relational one.
- It scales up easier compared to a relational database.
- MongoDB is a powerful, flexible, and scalable general-purpose database.
- It provides the following features:
  - Indexing
  - Aggregations
  - File Storage
  - Special collection types

# MongoDB Ease of Use

---

- The concept of a row is replaced with a document which is more flexible.
- By using documents and arrays, complex hierarchical relationships can be represented with a single record.
- MongoDB is schemaless.
- There is no predefined schema.
- The type and size of a document's keys and values can be variable.
- Add or remove fields is easier.
- Different models can be chosen

# MongoDB Easy Scaling

- As data grows at an incredible pace, the databases need to scale up.
- To scale:
  - Large machines can be used to scale up
    - Expensive
    - There is physical limit, more powerful machine may not exist.
  - Partitioning data across more machines can help scale out
    - More storage space by adding servers and computers to your cluster
      - Cheaper
      - But difficult to manage thousands of machines
  - MongoDB as a document-oriented model scales out easier by splitting data across multiple servers.

# MongoDB Basic Concepts

---

- Document
  - A document is the basic unit of data
  - A document is equivalent to a row in a relational database
- Collection
  - a collection can be considered as a table but with a dynamic schema
- One MongoDB instance can host multiple collections.

# Documents

---

- Every document has a special key (Id)
- The key of a document is unique within a collection
- Example:

```
{"greeting" : "Hello, world!"}
```
- Key: “greeting”
- Value: “hello, world!”
- A document can contain multiple key/value pairs:
- ```
{"greeting" : "Hello, world!", "foo" : 3}
```
- Key: “greeting” and value: “hello, world!”
- Key: “foo” and value: 3
- Notice that the type of these two values are different. One is integer and the other one is string.

# Document Key

- Document Key
- The type of a key is string (UTF-8 characters).
- The key cannot be the null terminator '\0'.
- Do not include \$ in a key.
- MongoDB is type-sensitive and case-sensitive:
  - { "foo" : 3 }
  - { "foo" : "3" }
- The above documents are distinct.
- The following documents are distinct as well
  - { "foo" : 3 }
  - { "Foo" : 3 }
- In Oracle the column empid and Empid are the same
- The columns names are not case-sensitive

# Documents

- Duplicate Keys
- A document cannot contain duplicate keys:
  - `{"greeting" : "Hello, world!", "greeting" : "Hello, MongoDB!"}`
  - The above document is not a legal document because it has duplicate keys.
  - Will cause an error
  - This is similar to having two identically named columns in a table row in a relational database

# Collections

- A collection is a group of documents.
- A collection in MongoDB can be considered as similar to a table in a relational database.
- The collection has a dynamic schema.
- Documents within a collection can have different schemas.
  - { "greeting" : "Hello, world!" }
  - { "foo" : 5 }
- The documents have different keys and values and value types
- Two different documents inside the same collection
- A document can be in any collection.
- Can't do the same thing with rows, columns and tables in a relational database

## Relational Database

### Customer Table

| Name         | Address          | Phone Number   |
|--------------|------------------|----------------|
| Todd Lynn    | 90 Park Pl.      | (374) 919-8909 |
| Margot Parks | 2 Sunset Dr.     | (252) 391-3585 |
| Ali Garcia   | 1902 Windsor St. | (204) 870-7819 |
| Susan Miller | 39 Kings Highway | (318) 553-7260 |

## Document Database

### Customer Collection

**Name:**  
Todd Lynn  
**Address:**  
90 Park Pl.  
**Phone Number:**  
(374) 919-8909

**Name:**  
Margot Parks  
**Address:**  
2 Sunset Dr.  
**Phone Number:**  
(252) 391-3585

**Name:**  
Ali Garcia  
**Address:**  
1902 Windsor St.  
**Phone Number:**  
(204) 870-7819

**Name:**  
Susan Miller  
**Address:**  
39 Kings Highway  
**Phone Number:**  
(318) 553-7260

# Nomenclatures

| SQL   | Table      | Row      | Join              |
|-------|------------|----------|-------------------|
| NOSQL | Collection | Document | Embedded Document |

# Lab

- Lab 7 number 3 asks for you to get started with MongoDB, so you can create a collection and documents based on your assigned database model.
- The MongoDB labs will be done on your pc's or laptops and will require some installation
- The MongodB labs will be demonstrated online since the schools machines are not set up for this new approach

- Set up an Atlas Account
- Set up Visual Studio Code
- Add the MongoDB extension to VS Code
- Connect from VS Code to Atlas
- Instructions for Atlas and VS Code

# Connection String

AutoSave (Off) Connection String • Last Modified: 13 March Search RP -

e Home Insert Draw Design Layout References Mailings Review View Help

## Connection String

mongoDB Atlas

`mongodb+srv://Russ_Pangborn:mypass@cluster0.dyoiaud.mongodb.net/test`

|           |          |               |
|-----------|----------|---------------|
| user name | password | database name |
|-----------|----------|---------------|

can be used to connect from the mongo shell, mongo DB compass, visual studio code or any other application

To locate the connection string in the atlas dashboard - Click on the database link on left hand side

To see the connection string, click on the connect button for cluster 0

You have options to connect to shell, application or MongoDB compass (you want compass)

1 of 1 75 words English (Canada) Text Predictions: On Accessibility: Good to go 100%

Firefox (19,823 unread) Log in | MongoDB Content Database Deploy X mongoDB atlas - DBS311 - Week 1 - Re Date() — MongoDB Connecting to a MongoDB Limits + - × https://cloud.mongodb.com/v2/636db0d46178235d3afad748#clusters/connect?clusterId=Cluster0

Getting Started ASUS Software Portal ... MyASUS Software - G... Seneca Blackboard Russ Pangborn Year 2022 Calendar - ...

Seneca Coll... Access Manager Project 0 Atlas

SENECA COLLEGE > PROJECTS Database I

Find a database deployment

DEPLOYMENT Database PREVIEW Data Lake DATA SERVICES Triggers Data API Data Federation SECURITY Quickstart Database Access Network Access Advanced New On Atlas

VERSION 5.0.13 REGION AWS / Other

Enhance Your Experience For production through richer metrics, upgrade to a dedicated cluster now! Upgrade

System Status: All Good

Connect to Cluster0

✓ Setup connection security ✓ Choose a connection method Connect

Connect Using VS Code

1 Install MongoDB for VS Code. In VS Code, open "Extensions" in the left navigation and search for "MongoDB for VS Code." Select the extension and click install.

2 In VS Code, open the Command Palette. Click on "View" and open "Command Palette." Search "MongoDB: Connect" on the Command Palette and click on "Connect with Connection String."

3 Connect to your MongoDB deployment. Paste your connection string into the Command Palette.

```
mongodb+srv://Russ_Pangborn:<password>@cluster0.dyoiaud.mongodb.net/test
```

Replace `<password>` with the password for the `Russ_Pangborn` user. When entering your password, make sure all special characters are URL encoded.

4 Click "Create New Playground" in MongoDB for VS Code to get started. Learn more about Playgrounds

Having trouble connecting? View our troubleshooting documentation

All Clusters Get Help Russell + Create FREE SHARED Data Size 155.3 KB Last 3 days 512.0 MB

mongodbsrv://Russ\_Pangborn:<password>@cluster0.dyoiaud.mongodb.net/test

mongodbsrv://Russ\_Pangborn:<password>@cluster0.dyoiaud.mongodb.net/test

# Temporary IP address when at campus 0.0.0.0

The screenshot shows the MongoDB Cloud interface for managing network access. The top navigation bar includes tabs for Courses, Network Access (which is active), and other sections like Getting Started, ASUS Software Portal, MyASUS Software, Pricing, Russ Pangborn, Year 2022 Calendar, Institution Page, and cubic meter to gallons conversion.

The main menu on the left lists Project 0, DATABASE, SERVICES, SECURITY, and Network Access (which is also highlighted in green). The DATABASE section is currently selected, showing sub-options like Clusters, Atlas Search, Stream Processing, Triggers, Migration, Data Federation, and Data API.

The central content area is titled "Network Access" and displays the "IP Access List". The table shows one entry:

| IP Address                                   | Comment | Status | Actions                                       |
|----------------------------------------------|---------|--------|-----------------------------------------------|
| 0.0.0.0/0 (includes your current IP address) |         | Active | <button>EDIT</button> <button>DELETE</button> |

A message above the table states: "You will only be able to connect to your cluster from the following list of IP Addresses:". Below the table, the system status is shown as "All Good".

At the bottom of the page, there are links for System Status, Status, Terms, Privacy, Atlas Blog, Contact Sales, and Advanced.

# MongoDB Group Database Collections

- Each group needs to select what their Lab database Collections are going to be modeled on
- Tired of books, movies, games as student selections. I am doing audiobooks mainly.
- Email me a list starting from most desirable to least desirable
- I will go in order of emails arrival and inform the group what their databases should model
- If I didn't award your first item – someone beat you to it.

# List For your MongoDB

---

- 1 Lumber
- 2 Flowers
- 3 Guns
- 4 Health Foods
- 5 Tea
- 6 Office Furniture
- 7 Dishwashers
- 8 Tools
- 9 Winter Coats

# Lab 7 - 10

---

- Lab 7 will be marked in class
- Labs 8 – 10 will be run online (not in the classroom)  
Lectures still on campus
- The Mongo Labs 8 – 10 will be group labs using your assignment group.

# Remaining Schedule

| Week      | Lecture                               | Lab                          |
|-----------|---------------------------------------|------------------------------|
| Nov 3 WK  | Lecture on Cursors,Functions, MongoDB | Lab 6 due                    |
| Nov 10 WK | MongoDB Query                         | Oracle Lab 7 due             |
| Nov 17 WK | MongoDB Update Documents              | Visual Code Lab 8 Due online |
| Nov 24 WK | Aggregation                           | Visual Code Lab 9 due online |
| Dec 1 WK  | Review for Final                      | Assign2 due online           |
| Dec 8 WK  | Final Test                            | Lab 10 due online            |