Mu Sigma

Database & SQL

Induction Training

Do The Math

Chicago, IL
Bangalore, India
www.mu-sigma.com

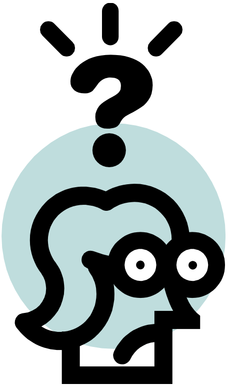
January 16, 2013

Welcome to SQL & Database Training

So… what is SQL and why do we need it?

SQL stands for:

Scarcely Qualified Language



Just kidding!

So… what is SQL and why do we need it?

SQL stands for:
Structured Query Language

That’s all you will learn for now!

Kidding!!... But we will hold off SQL for sometime

Let us spend two minutes on this…

What are your expectations from this training?

Before we dive into SQL, let’s talk about DATA…

Talk about me?

Sorry… when we say DATA,
we mean **information**

Let’s try and understand where and how data is created

A Retail Supermarket

A Customer

What kind of information is captured?

A retail store captures a lot of information regarding a customer’s shopping behavior

Who is coming to my store?

What products are being bought?

What frequently are my products purchased?

Let’s list out some metrics or data a retail store might collect

1 **Customer Information**

- Customer Name
- Customer Address
- Age
- Telephone

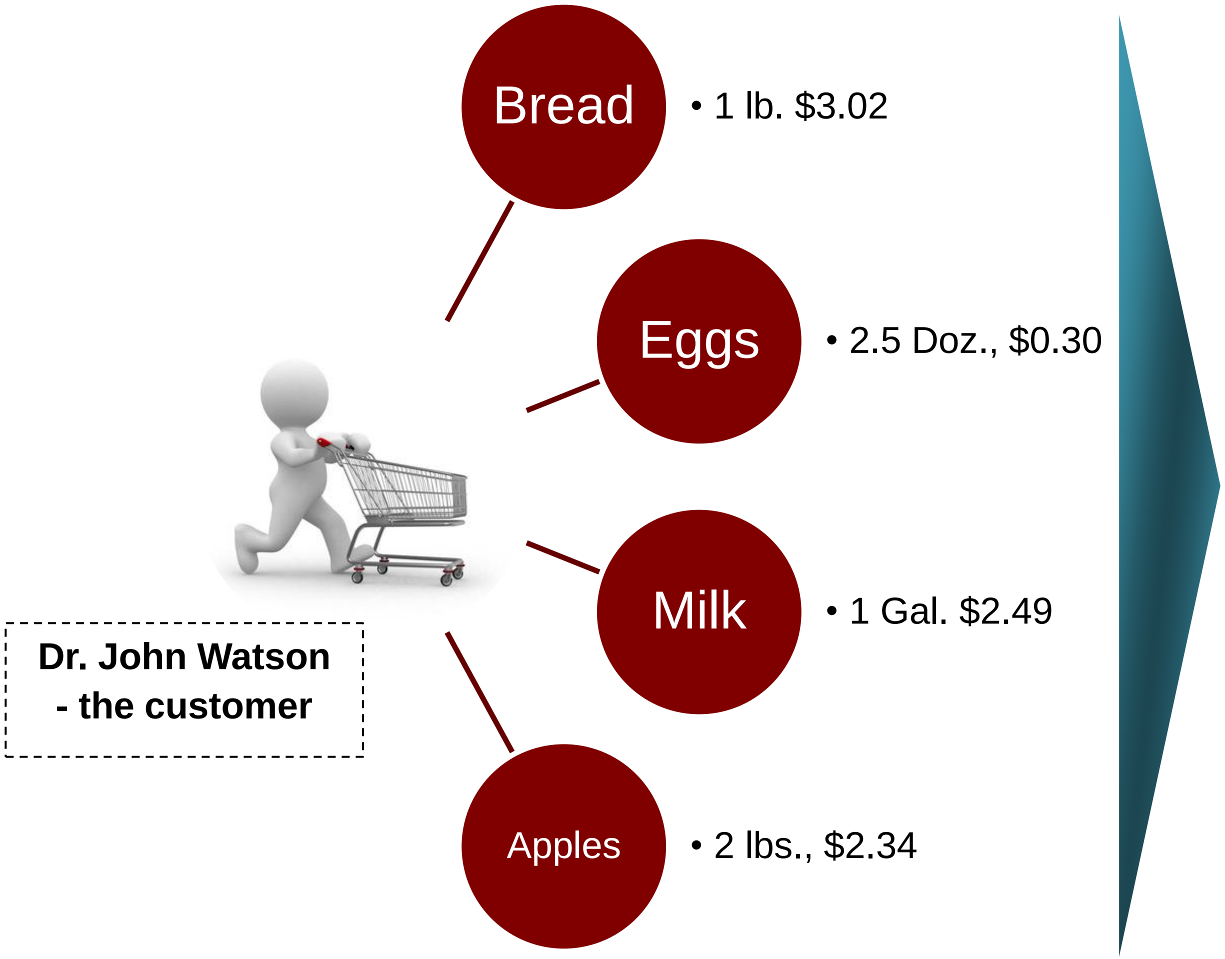
2 **Product Information**

- Product Name
- Product Class
- Price
- Discount

2 **Transaction Information**

- Date
- Customer Name
- Product Name
- Cost

Now, consider purchases of a single customer



Bread

- 1 lb. $3.02

Eggs

- 2.5 Doz., $0.30

Milk

- 1 Gal. $2.49

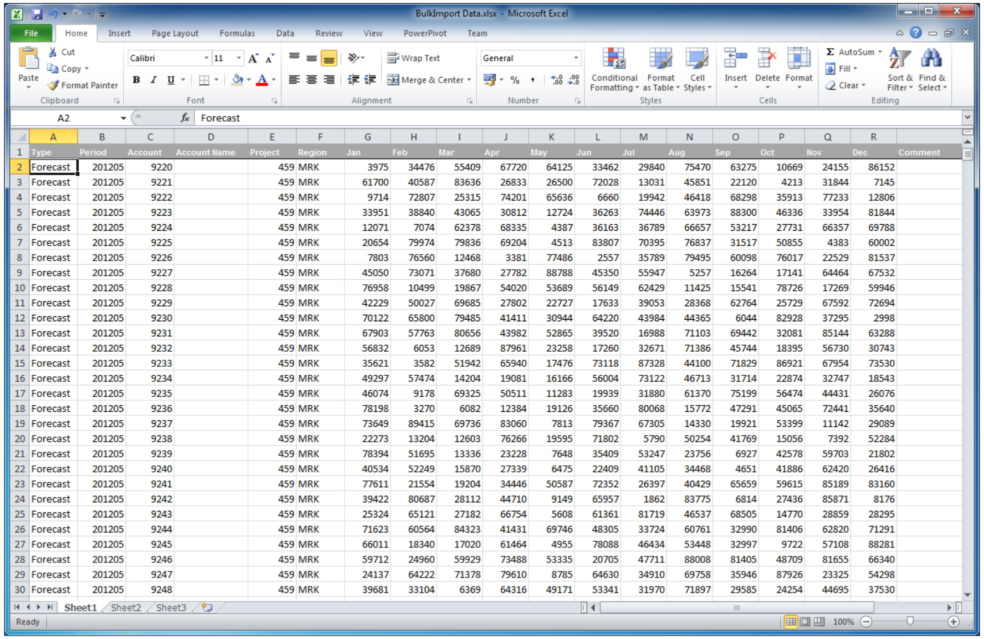
Apples

- 2 lbs., $2.34

Dr. John Watson - the customer

How do we store this information?

What if we use Excel to store this information?



Type	Period	Account	Account Name	Project	Region	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Comment
Forecast	201205	9220		459	MRK	3975	34476	55409	67720	64125	33462	29840	75470	63275	10669	24155	86152	
Forecast	201205	9221		459	MRK	61700	40587	83636	26833	26500	72028	13031	45851	22120	4213	31844	7145	
Forecast	201205	9222		459	MRK	9714	72807	25315	74201	65636	6660	19942	46418	68298	35913	77233	12806	
Forecast	201205	9223		459	MRK	33951	38840	43065	30812	12724	36263	74446	63973	88300	46336	33954	81844	
Forecast	201205	9224		459	MRK	12071	7074	62378	68335	4387	36163	36789	66657	53217	27731	66357	69788	
Forecast	201205	9225		459	MRK	20654	79974	79836	69204	4513	83807	70395	76837	31517	50855	4383	60002	
Forecast	201205	9226		459	MRK	7803	76560	12468	3381	77486	2557	35789	79495	60098	76017	22529	81537	
Forecast	201205	9227		459	MRK	45050	73071	37680	27782	88788	45350	55947	5257	16264	17141	64464	67532	
Forecast	201205	9228		459	MRK	76958	10499	19867	54020	53689	56149	62429	11425	15541	78726	17269	59946	
Forecast	201205	9229		459	MRK	42229	50027	69685	27802	22727	17633	39053	28368	62764	25729	67592	72694	
Forecast	201205	9230		459	MRK	70122	65800	79485	41411	30944	64220	43984	44365	6044	82928	37295	2998	
Forecast	201205	9231		459	MRK	67903	57763	80656	43982	52865	39520	16988	71103	69442	32081	85144	63288	
Forecast	201205	9232		459	MRK	56832	6053	12689	87961	23258	17260	32671	71386	45744	18395	56730	30743	
Forecast	201205	9233		459	MRK	35621	3582	51942	65940	17476	73118	87328	44100	71829	86921	67954	73530	
Forecast	201205	9234		459	MRK	49297	57474	14204	19081	16166	56004	73122	46713	31714	22874	32747	18543	
Forecast	201205	9235		459	MRK	46074	9178	69325	50511	11283	19939	31880	61370	75199	56474	44431	26076	
Forecast	201205	9236		459	MRK	78198	3270	6082	12384	19126	35660	80068	15772	47291	45065	72441	35640	
Forecast	201205	9237		459	MRK	73649	89415	69736	83060	7813	79367	67305	14330	19921	53399	11142	29089	
Forecast	201205	9238		459	MRK	22273	13204	12603	76266	19595	71802	5790	50254	41769	15056	7392	52284	
Forecast	201205	9239		459	MRK	78394	51695	13336	23228	7648	35409	53247	23756	6927	42578	59703	21802	
Forecast	201205	9240		459	MRK	40534	52249	15870	27339	6475	22409	41105	34468	4651	41886	62420	26416	
Forecast	201205	9241		459	MRK	77611	21554	19204	34446	50587	72352	26397	40429	65659	59615	85189	83160	
Forecast	201205	9242		459	MRK	39422	80687	28112	44710	9149	65957	1862	83775	6814	27436	85871	8176	
Forecast	201205	9243		459	MRK	25324	65121	27182	66754	5608	61361	81719	46537	68505	14770	28859	28295	
Forecast	201205	9244		459	MRK	71623	60564	84323	41431	69746	48305	33724	60761	32990	81406	62820	71291	
Forecast	201205	9245		459	MRK	66011	18340	17020	61464	4955	78088	46434	53448	32997	9722	57108	88281	
Forecast	201205	9246		459	MRK	59712	24960	59929	73488	53335	20705	47711	88008	81405	48709	81655	66340	
Forecast	201205	9247		459	MRK	24137	64222	71378	79610	8785	64630	34910	69758	35946	87926	23325	54298	
Forecast	201205	9248		459	MRK	39681	33104	6369	64316	49171	53341	31970	71897	29585	24254	44695	37530	

Excel Spreadsheet

- What if 100 people want to modify the file?
- What if the number of transactions cross a million?
- How do you manage a million pieces of information?
- **An Excel document is clearly not the right choice for storing information in a transactional environment**
- **What we need is a file system which can**
  - **Hold multiple tables of information**
  - **Do not run out of space**
  - **Allow multiple people to access and modify the information simultaneously**

A database allows multiple users to interact with the same piece of information simultaneously

InterBase Firebird Panorama

SQLBase **SQL Server** Sybase

Filemaker **RDBMS** Access

FoxPro **Oracle DB2** SQLite

PostGreSQL

MaxDB **MySQL** Teradata

- More on **RDBMS** in the later slides …

How should we store every transaction in the database?

Store every transaction as a single row

Date	First Name	Last Name	Bread	Eggs	Milk	Apple
3-Mar-12	John	Watson	$3.02	$0.30	$2.49	$2.34

Advantages

- I can find everything about a purchase by looking at single row

Disadvantages

- Not feasible as the number of columns is undefined (if a customer buys 20 items)

How should we store every transaction in the database?

Store every item in a transaction as a single row

Date	First Name	Last Name	Product	Quantity	Price
3-Mar-12	John	Watson	Bread	1 lb.	$3.02
3-Mar-12	John	Watson	Eggs	2.4 doz.	$0.30
3-Mar-12	John	Watson	Milk	1 gal.	$2.49
3-Mar-12	John	Watson	Apple	2 lbs.	$2.34

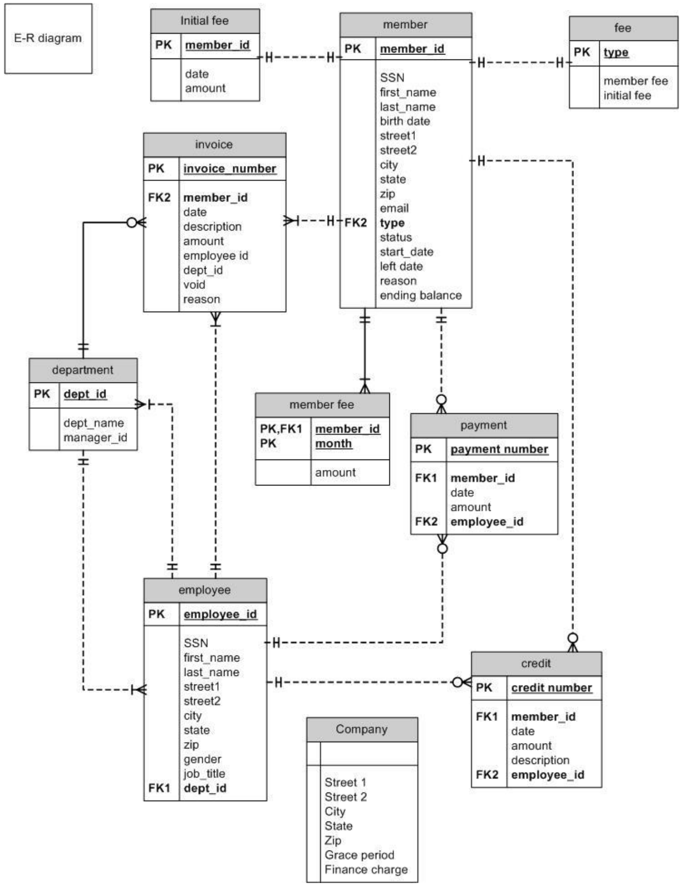
Advantages

- I don’t have to worry about how many items a customer is buying

Disadvantages

- There is duplication of information across the rows

That brings us to the subject of a relational database system



E-R diagram

Initial fee
PK member_id
date
amount

member
PK member_id
SSN
first_name
last_name
birth date
street1
street2
city
state
zip
email
FK2 type
status
start_date
left date
reason
ending balance

fee
PK type
member fee
initial fee

invoice
PK invoice_number
FK2 member_id
date
description
amount
employee id
dept_id
void
reason

department
PK dept_id
dept_name
manager_id

member fee
PK,FK1 member_id
PK month
amount

payment
PK payment number
FK1 member_id
date
amount
FK2 employee_id

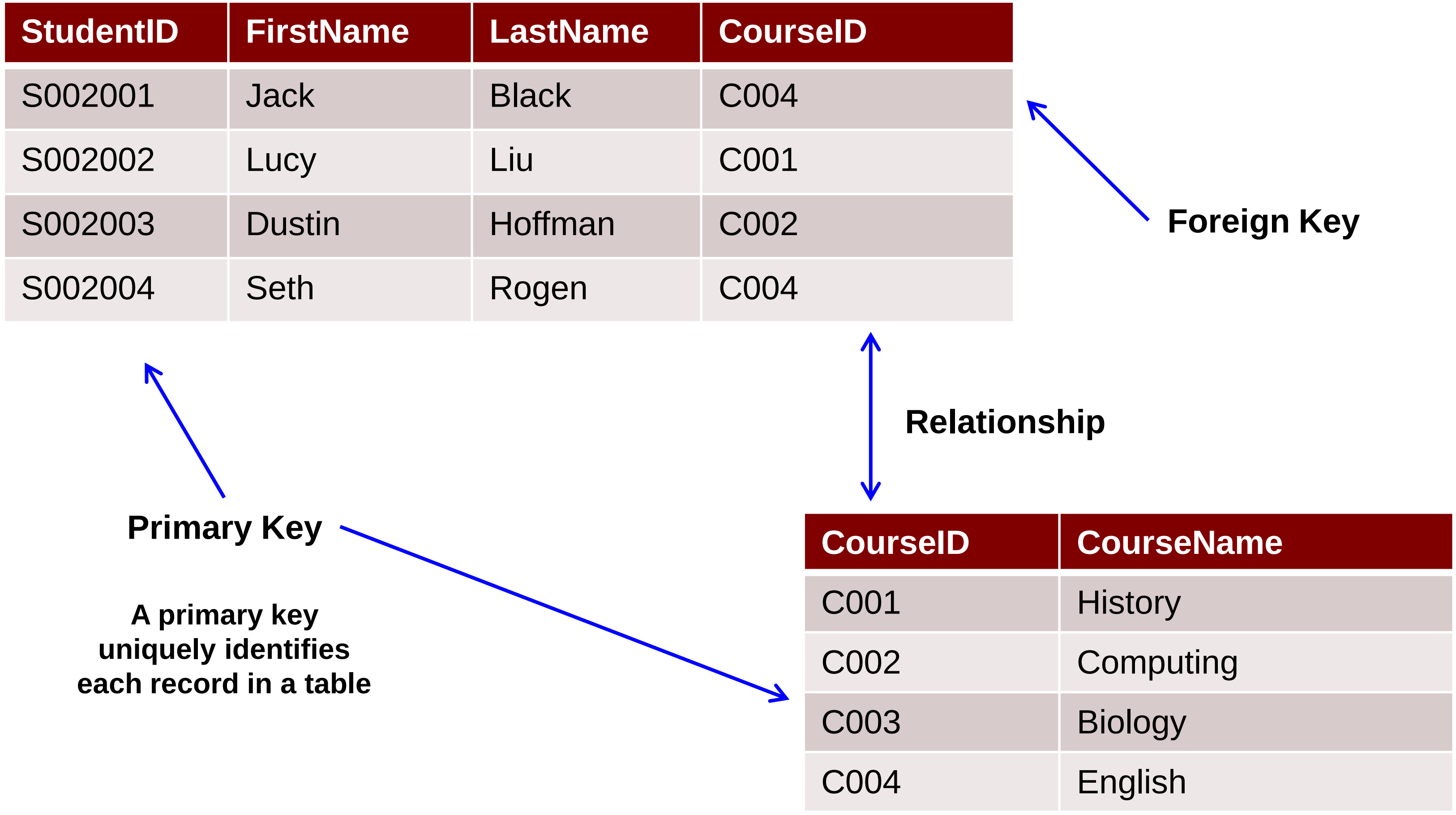
employee
PK employee_id
SSN
first_name
last_name
street1
street2
city
state
zip
gender
job_title
FK1 dept_id

credit
PK credit number
FK1 member_id
date
amount
description
FK2 employee_id

Company
Street 1
Street 2
City
State
Zip
Grace period
Finance charge

- A database is just a location to store and retrieve data
- A relational database is one which treats all of its data as a collection of relations
- Each table in a relational database has unique key (also referred to as the primary key)
- This primary key can also be present in another table as a foreign key and in process creates a relation between two tables

To create a relational database, we need to understand the granularity or level of each table



StudentID	FirstName	LastName	CourseID
S002001	Jack	Black	C004
S002002	Lucy	Liu	C001
S002003	Dustin	Hoffman	C002
S002004	Seth	Rogen	C004

Foreign Key

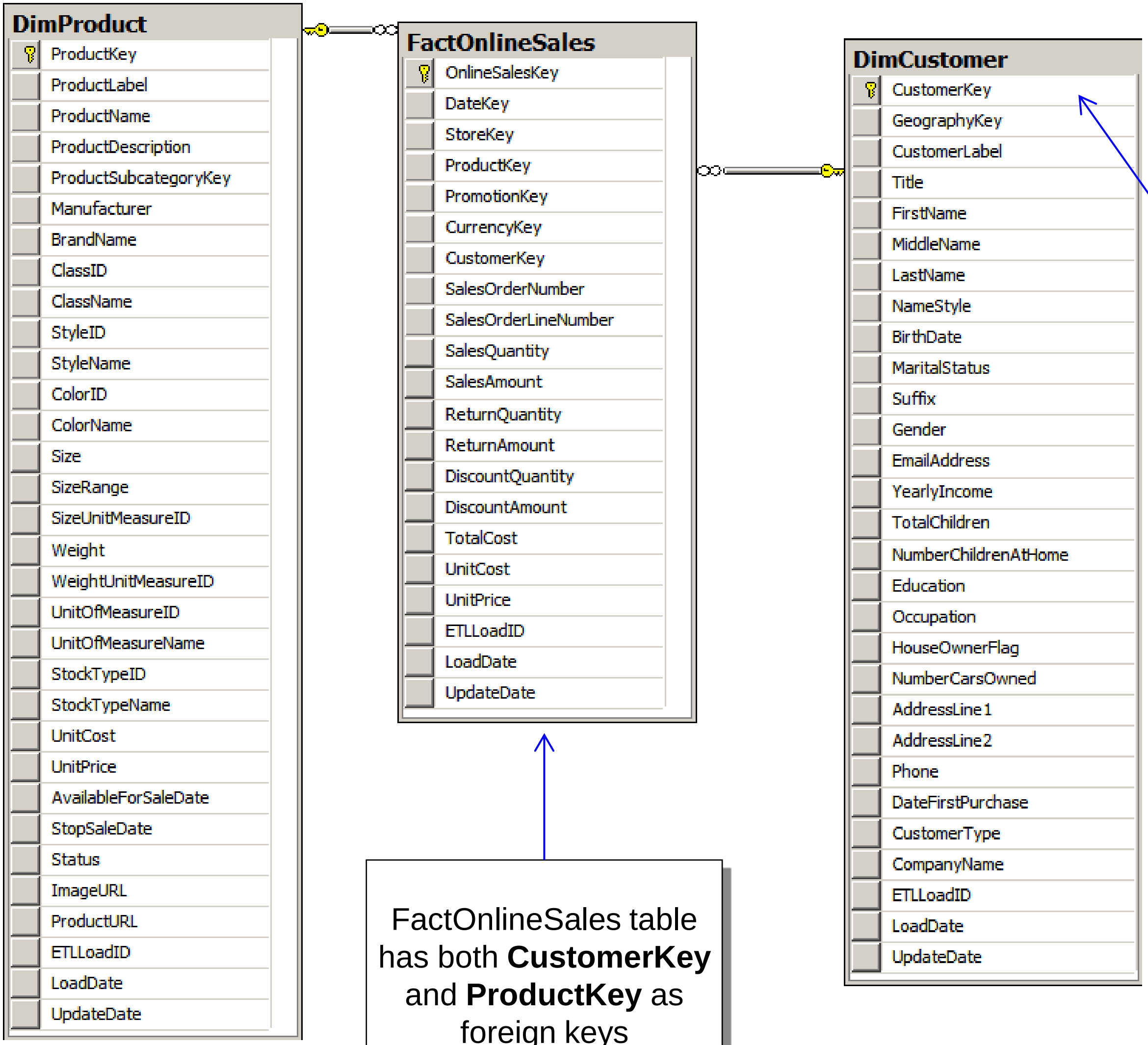
Relationship

Primary Key

A primary key uniquely identifies each record in a table

CourseID	CourseName
C001	History
C002	Computing
C003	Biology
C004	English

A simple relational database



DimProduct

- ProductKey
- ProductLabel
- ProductName
- ProductDescription
- ProductSubcategoryKey
- Manufacturer
- BrandName
- ClassID
- ClassName
- StyleID
- StyleName
- ColorID
- ColorName
- Size
- SizeRange
- SizeUnitMeasureID
- Weight
- WeightUnitMeasureID
- UnitOfMeasureID
- UnitOfMeasureName
- StockTypeID
- StockTypeName
- UnitCost
- UnitPrice
- AvailableForSaleDate
- StopSaleDate
- Status
- ImageURL
- ProductURL
- ETLLoadID
- LoadDate
- UpdateDate

FactOnlineSales

- OnlineSalesKey
- DateKey
- StoreKey
- ProductKey
- PromotionKey
- CurrencyKey
- CustomerKey
- SalesOrderNumber
- SalesOrderLineNumber
- SalesQuantity
- SalesAmount
- ReturnQuantity
- ReturnAmount
- DiscountQuantity
- DiscountAmount
- TotalCost
- UnitCost
- UnitPrice
- ETLLoadID
- LoadDate
- UpdateDate

DimCustomer

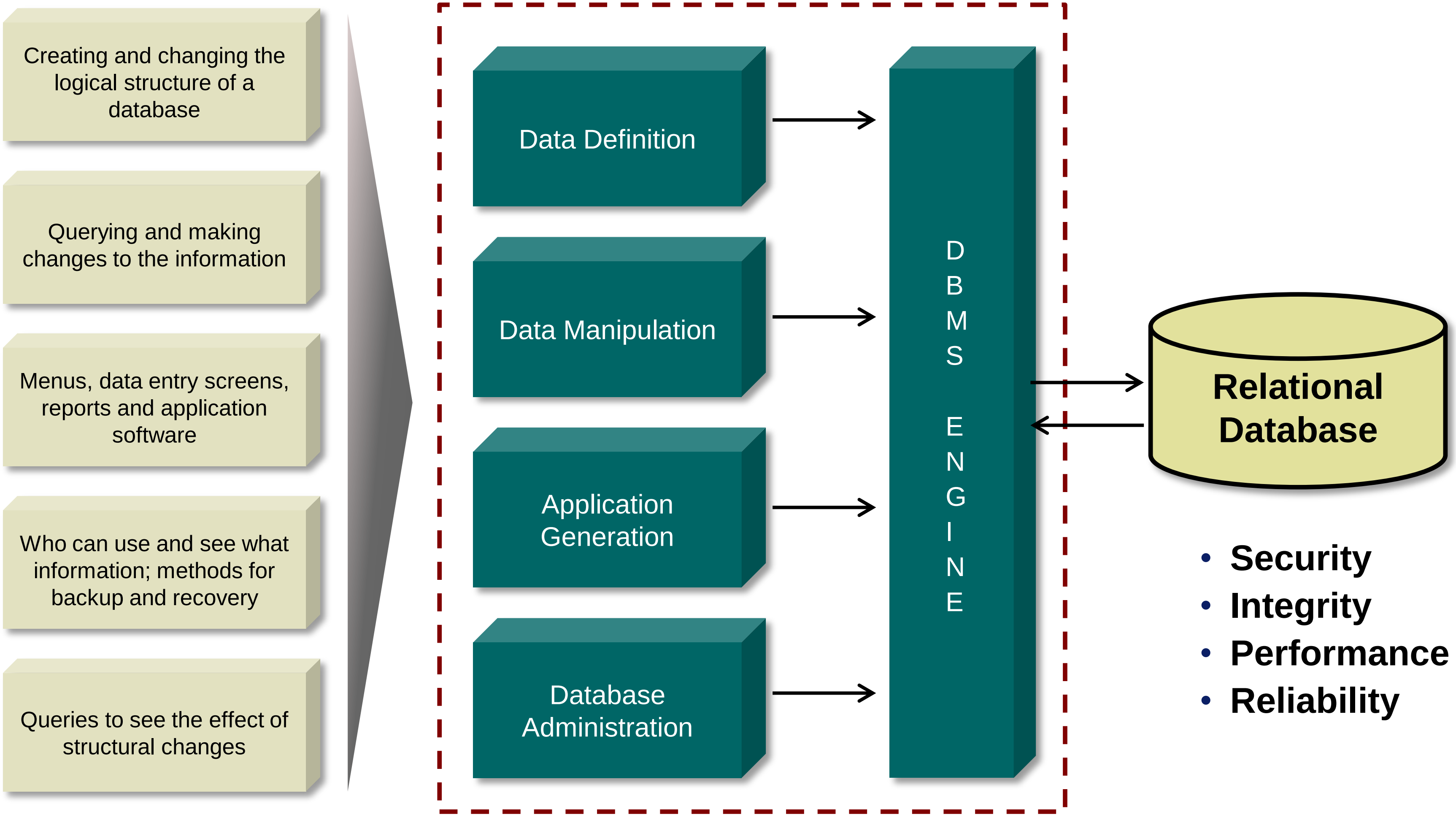
- CustomerKey
- GeographyKey
- CustomerLabel
- Title
- FirstName
- MiddleName
- LastName
- NameStyle
- BirthDate
- MaritalStatus
- Suffix
- Gender
- EmailAddress
- YearlyIncome
- TotalChildren
- NumberChildrenAtHome
- Education
- Occupation
- HouseOwnerFlag
- NumberCarsOwned
- AddressLine1
- AddressLine2
- Phone
- DateFirstPurchase
- CustomerType
- CompanyName
- ETLLoadID
- LoadDate
- UpdateDate

Product table has **ProductKey** as the primary key

Customer table has **CustomerKey** as the primary key

FactOnlineSales table has both **CustomerKey** and **ProductKey** as foreign keys

A database management system is not just a place to dump data

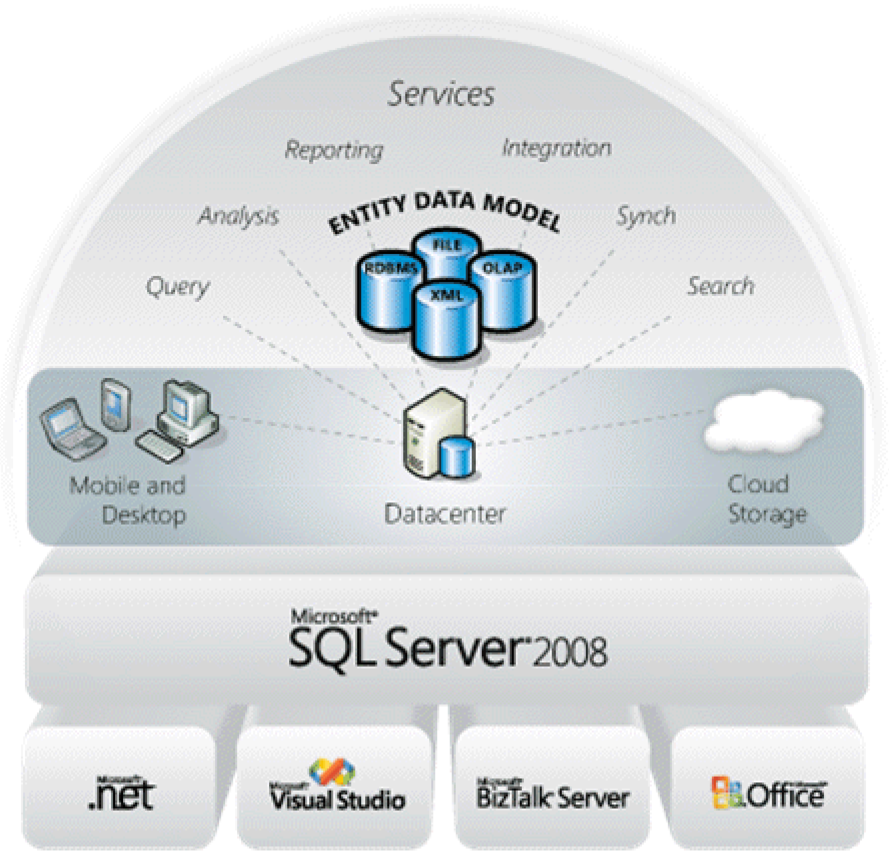

Creating and changing the logical structure of a database
Querying and making changes to the information
Menus, data entry screens, reports and application software
Who can use and see what information; methods for backup and recovery
Queries to see the effect of structural changes
Data Definition
Data Manipulation
Application Generation
Database Administration
DBMS ENGINE
Relational Database

- **Security**
- **Integrity**
- **Performance**
- **Reliability**

Coming back to what is SQL and why do we need it?

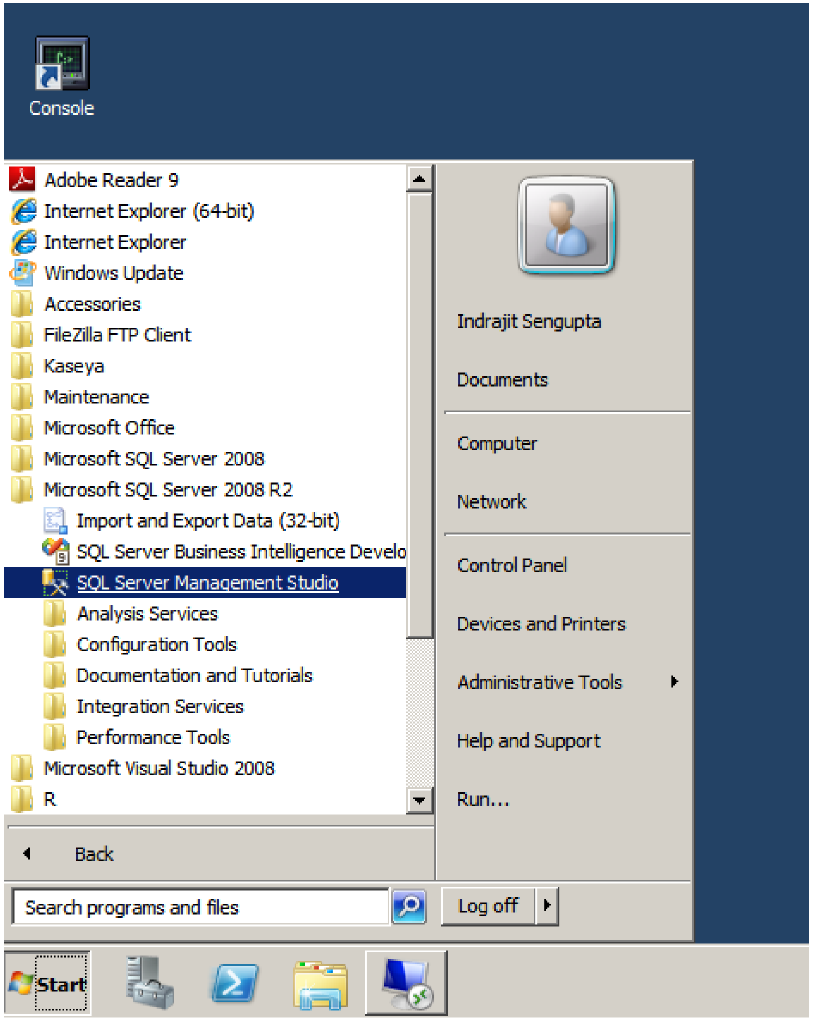
- SQL is the standard language for communicating with relational database management systems
- SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database
- Although most database systems use SQL, most of them also have their own additional proprietary extensions that are usually only used on their system
- However the standard SQL commands can be used to accomplish almost everything one needs

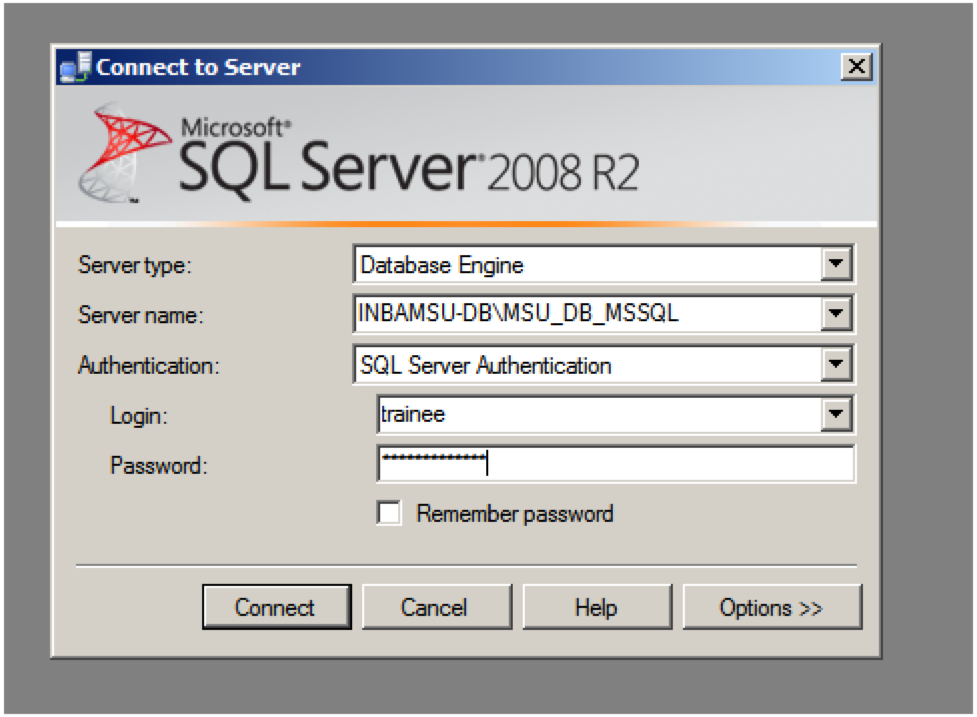
For our learning purpose we will be using the Microsoft SQL Server 2008 Express edition



- SQL Server 2008 R2 Express edition is a free version of SQL Server 2008 R2
- Almost everything that can be done in the commercial version is supported in the express edition
- Database size is restricted to 10 GB in size
- Uses only a single processor
- Uses only 1 GB of internal memory

Start up SQL Server Management Studio from your Start menu





Server type: Database Engine

Server name: INBAMSU-DB\MSU_DB_MSSQL

Authentication: SQL Server Authentication

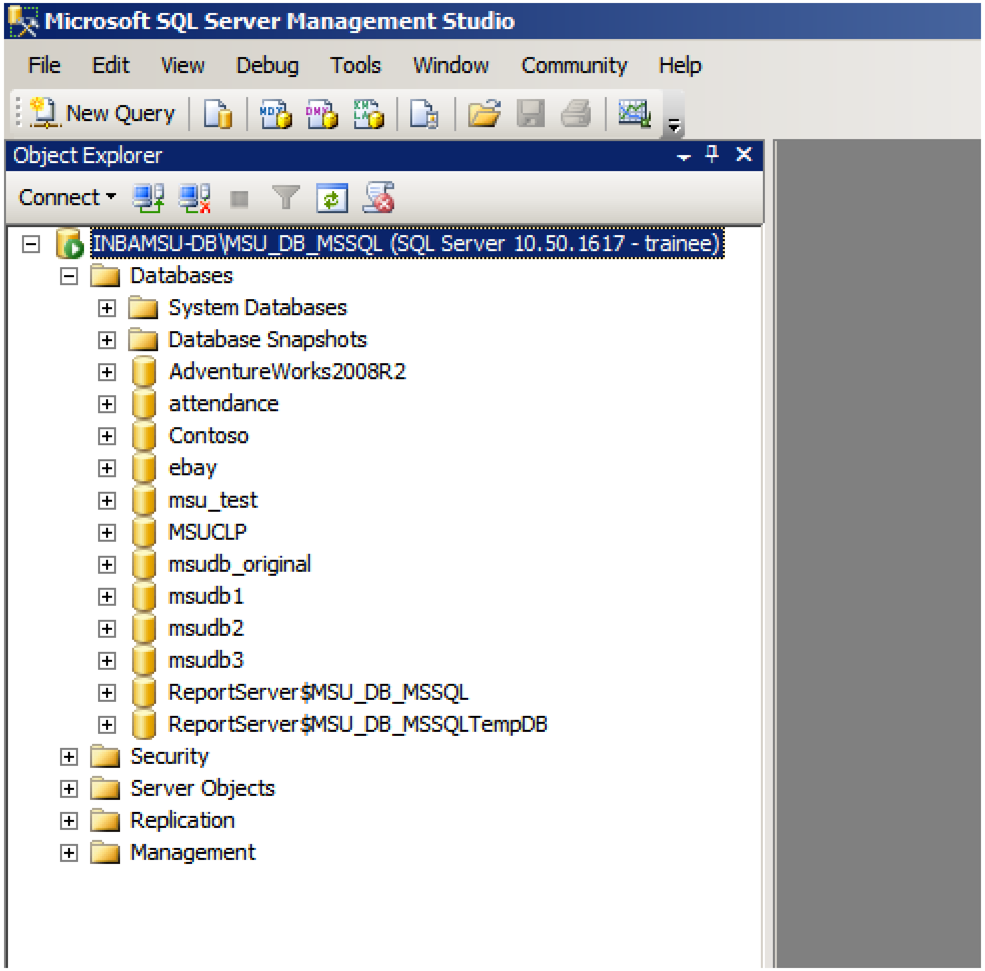
Login: trainee

Password: **************

□ Remember password

- Enter the server name: **INBAMSU-DB\MSU_DB_MSSQL**
- Enter user id: **trainee**
- Enter password: **password@1234**
- Remember to choose **SQL Server Authentication**

The Object Explorer window will list out all the databases


Microsoft SQL Server Management Studio
File Edit View Debug Tools Window Community Help
New Query
Object Explorer
Connect

- INBAMSU-DB\MSU_DB_MSSQL (SQL Server 10.50.1617 - trainee)
  - Databases
    - System Databases
    - Database Snapshots
    - AdventureWorks2008R2
    - attendance
    - Contoso
    - ebay
    - msu_test
    - MSUCLP
    - msudb_original
    - msudb1
    - msudb2
    - msudb3
    - ReportServer$MSU_DB_MSSQL
    - ReportServer$MSU_DB_MSSQLTempDB
  - Security
  - Server Objects
  - Replication
  - Management

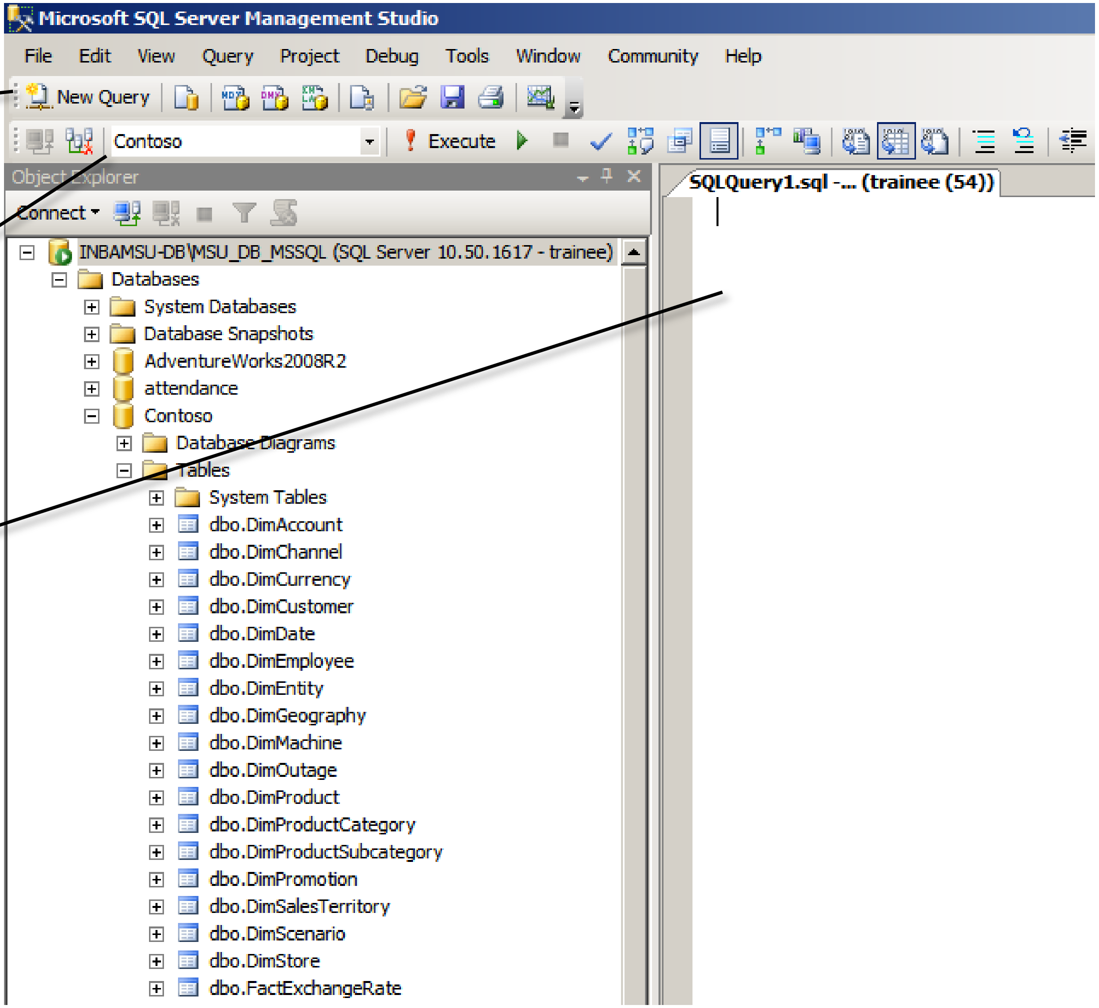
- Once you login to the server, the object explorer window on the left will list out all the objects
- Expand the **Database** object to display all the databases currently installed
- We will focus on the **Contoso** database for all our practice sessions

Open a query window by clicking on the New Query button

Open query window by clicking here

Select **Contoso** database from the drop down

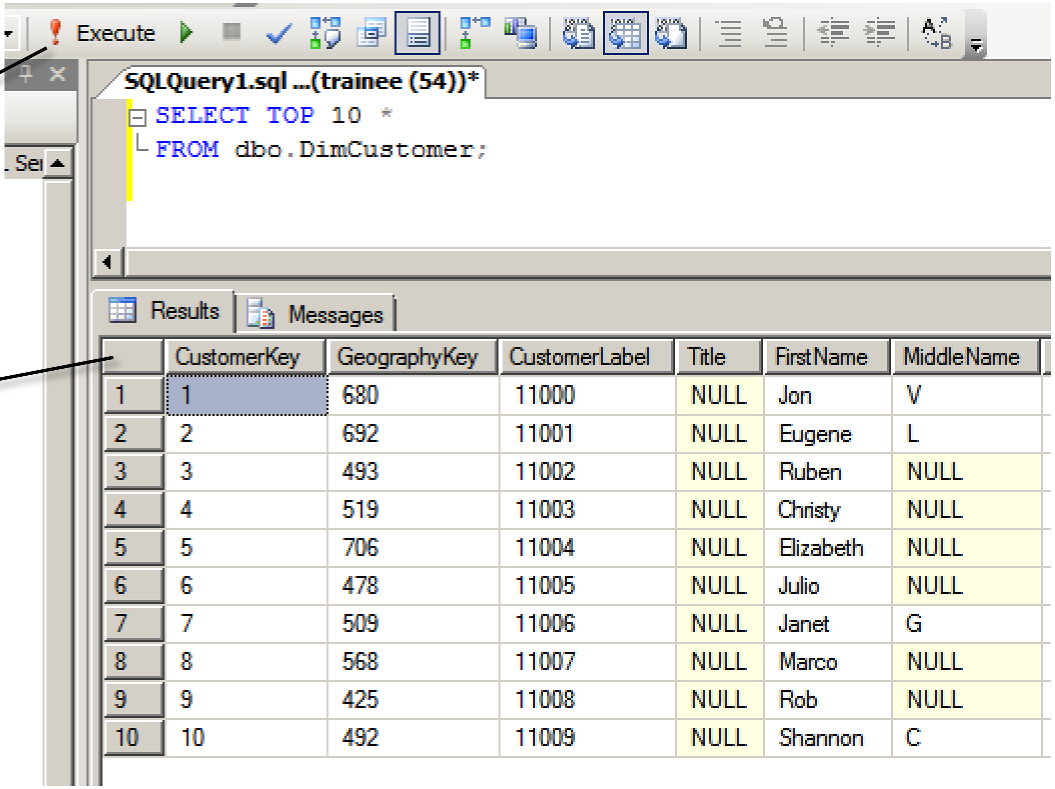
Enter all your queries here



Your first SQL query

Enter the query and press **F5** or **Execute** to submit the query

Query **output**

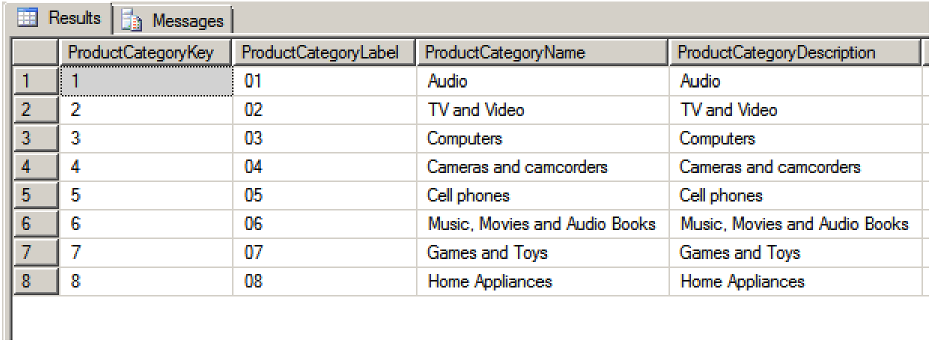


```sql
SELECT TOP 10 *
FROM dbo.DimCustomer;
```

	CustomerKey	GeographyKey	CustomerLabel	Title	FirstName	MiddleName
1	1	680	11000	NULL	Jon	V
2	2	692	11001	NULL	Eugene	L
3	3	493	11002	NULL	Ruben	NULL
4	4	519	11003	NULL	Christy	NULL
5	5	706	11004	NULL	Elizabeth	NULL
6	6	478	11005	NULL	Julio	NULL
7	7	509	11006	NULL	Janet	G
8	8	568	11007	NULL	Marco	NULL
9	9	425	11008	NULL	Rob	NULL
10	10	492	11009	NULL	Shannon	C

The SELECT query extracts rows from a table

```
SELECT *
FROM dbo.DimProductCategory;
```

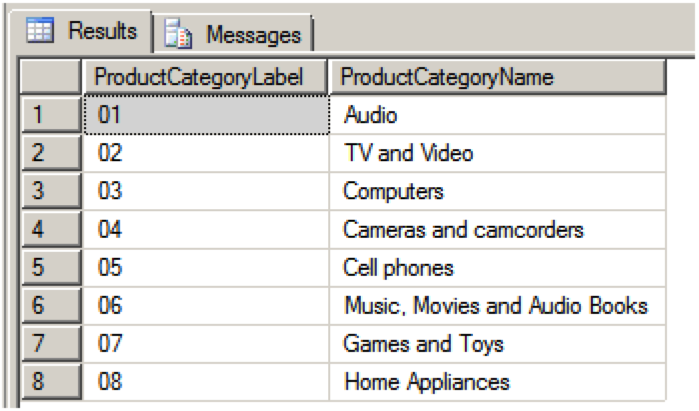


	ProductCategoryKey	ProductCategoryLabel	ProductCategoryName	ProductCategoryDescription
1	1	01	Audio	Audio
2	2	02	TV and Video	TV and Video
3	3	03	Computers	Computers
4	4	04	Cameras and camcorders	Cameras and camcorders
5	5	05	Cell phones	Cell phones
6	6	06	Music, Movies and Audio Books	Music, Movies and Audio Books
7	7	07	Games and Toys	Games and Toys
8	8	08	Home Appliances	Home Appliances

- The **SELECT** query is used to select rows from a table in a database
- The asterisk (*) indicates all rows and all columns
- The **FROM** statement indicates the table you want to access

You can specify the column names to focus on only a few fields

```
SELECT
ProductCategoryLabel,
ProductCategoryName
FROM dbo.DimProductCategory;
```

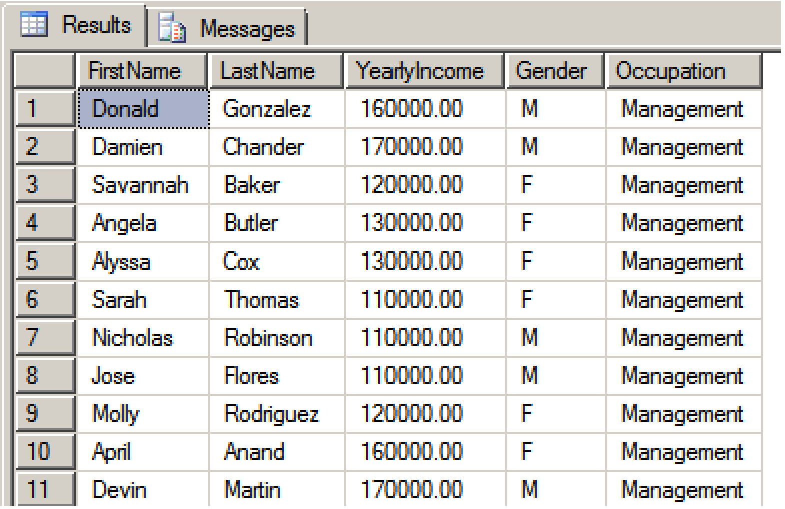


	ProductCategoryLabel	ProductCategoryName
1	01	Audio
2	02	TV and Video
3	03	Computers
4	04	Cameras and camcorders
5	05	Cell phones
6	06	Music, Movies and Audio Books
7	07	Games and Toys
8	08	Home Appliances

- You can specify the columns you want to display by listing out the names separated by a comma
- SQL is not case sensitive, hence lower case syntax is also correct

Use the WHERE statement to apply conditions on rows

```
SELECT
        FirstName,
        LastName,
        YearlyIncome,
        Gender,
        Occupation
FROM dbo.DimCustomer
WHERE YearlyIncome > 100000;
```

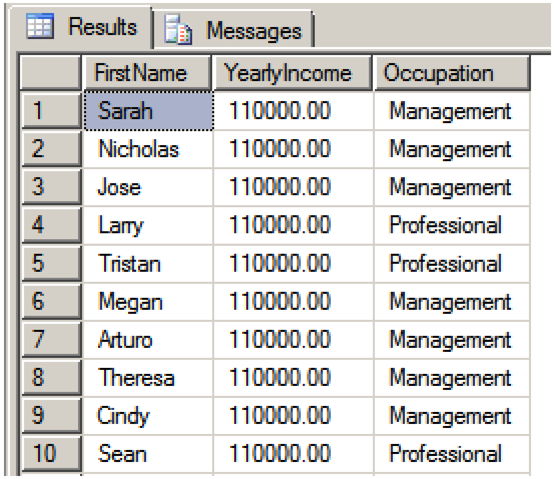


	FirstName	LastName	YearlyIncome	Gender	Occupation
1	Donald	Gonzalez	160000.00	M	Management
2	Damien	Chander	170000.00	M	Management
3	Savannah	Baker	120000.00	F	Management
4	Angela	Butler	130000.00	F	Management
5	Alyssa	Cox	130000.00	F	Management
6	Sarah	Thomas	110000.00	F	Management
7	Nicholas	Robinson	110000.00	M	Management
8	Jose	Flores	110000.00	M	Management
9	Molly	Rodriguez	120000.00	F	Management
10	April	Anand	160000.00	F	Management
11	Devin	Martin	170000.00	M	Management

- You can specify multiple conditions by separating each condition by the keywords **AND** or **OR**
- For better readability enclose the conditions with brackets

The ORDER statement sorts the output in ascending or descending order of a variable

```
SELECT
        FirstName,
        YearlyIncome,
        Occupation
FROM dbo.DimCustomer
WHERE YearlyIncome > 100000
ORDER BY YearlyIncome;
```

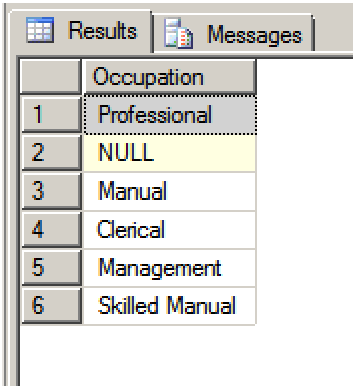


	FirstName	YearlyIncome	Occupation
1	Sarah	110000.00	Management
2	Nicholas	110000.00	Management
3	Jose	110000.00	Management
4	Larry	110000.00	Professional
5	Tristan	110000.00	Professional
6	Megan	110000.00	Management
7	Arturo	110000.00	Management
8	Theresa	110000.00	Management
9	Cindy	110000.00	Management
10	Sean	110000.00	Professional

- The default sort order is ascending – to sort in descending order, use the keyword **DESC** after the variable name

The DISTINCT keyword identifies unique rows from a table

```
SELECT DISTINCT
        Occupation
FROM dbo.DimCustomer;
```

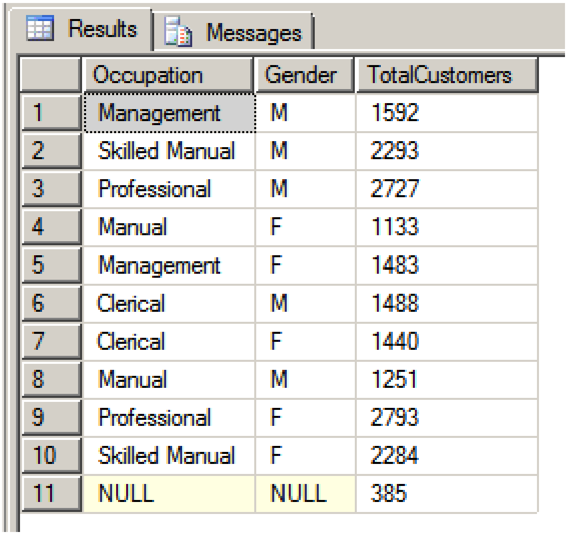


	Occupation
1	Professional
2	NULL
3	Manual
4	Clerical
5	Management
6	Skilled Manual

- Specifying multiple column names with **DISTINCT** keyword results in selecting unique combinations of all the columns from the table

The GROUP BY statement is used with summary functions to roll up tables

```
SELECT
        Occupation,
        Gender,
        COUNT(CustomerKey) AS
TotalCustomers
FROM dbo.DimCustomer
GROUP BY
        Occupation,
        Gender;
```

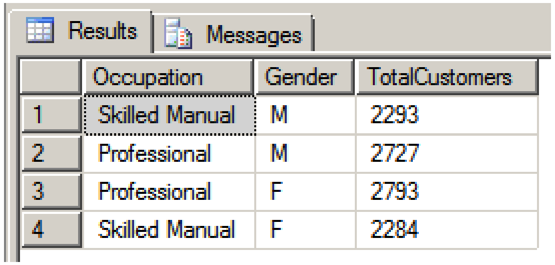


	Occupation	Gender	TotalCustomers
1	Management	M	1592
2	Skilled Manual	M	2293
3	Professional	M	2727
4	Manual	F	1133
5	Management	F	1483
6	Clerical	M	1488
7	Clerical	F	1440
8	Manual	M	1251
9	Professional	F	2793
10	Skilled Manual	F	2284
11	NULL	NULL	385

- Various summary functions are available in SQL language like **SUM**, **AVG**, **MIN**, **MAX**, **STDEV**
- Additional functions may be available which can be specific to the database you are using
- A simple **COUNT(*)** will count the number of rows

The HAVING statement applies conditions on a summary variable

```
SELECT
        Occupation,
        Gender,
        COUNT(CustomerKey) AS
        TotalCustomers
FROM dbo.DimCustomer
GROUP BY
        Occupation, Gender
HAVING COUNT(CustomerKey) >
        2000
```

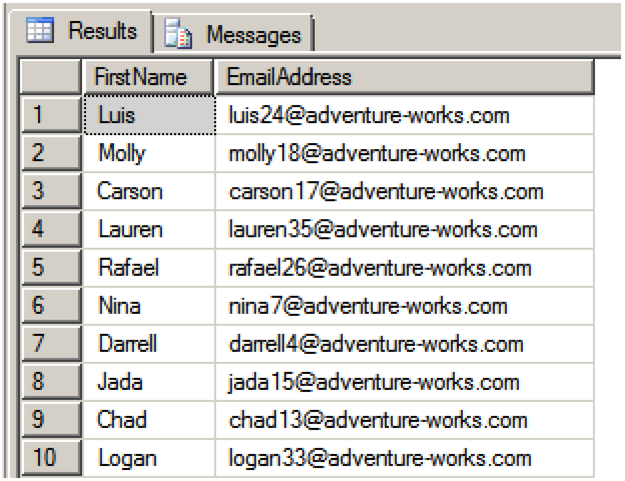


	Occupation	Gender	TotalCustomers
1	Skilled Manual	M	2293
2	Professional	M	2727
3	Professional	F	2793
4	Skilled Manual	F	2284

- **HAVING** clause works on aggregation of data rather than the rows of the data prior to the aggregation

A **SUBQUERY** is a query on the data that is found within another query statement

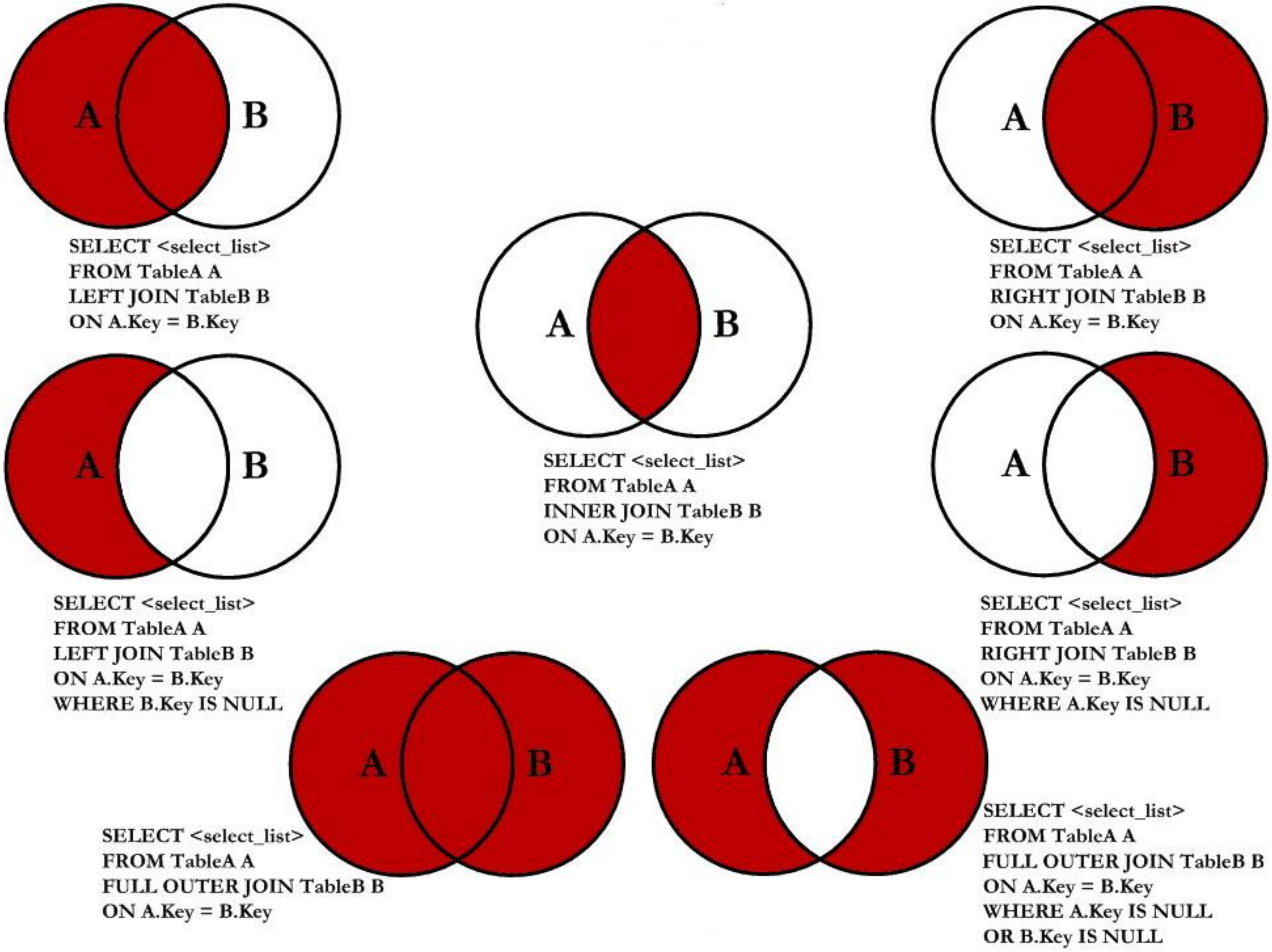
```
SELECT
        FirstName,
        EmailAddress
FROM dbo.DimCustomer
WHERE GeographyKey IN (
        SELECT GeographyKey
        FROM dbo.DimGeography
        WHERE  CityName =
        ‘Beverley Hills’);
```



	FirstName	EmailAddress
1	Luis	luis24@adventure-works.com
2	Molly	molly18@adventure-works.com
3	Carson	carson17@adventure-works.com
4	Lauren	lauren35@adventure-works.com
5	Rafael	rafael26@adventure-works.com
6	Nina	nina7@adventure-works.com
7	Darrell	darrell4@adventure-works.com
8	Jada	jada15@adventure-works.com
9	Chad	chad13@adventure-works.com
10	Logan	logan33@adventure-works.com

- The sub query can only return one column at a time and the inner query gets executed before the outer one

JOIN keyword is used to query data from two or more tables based on a relationships between the columns in these tables.



```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

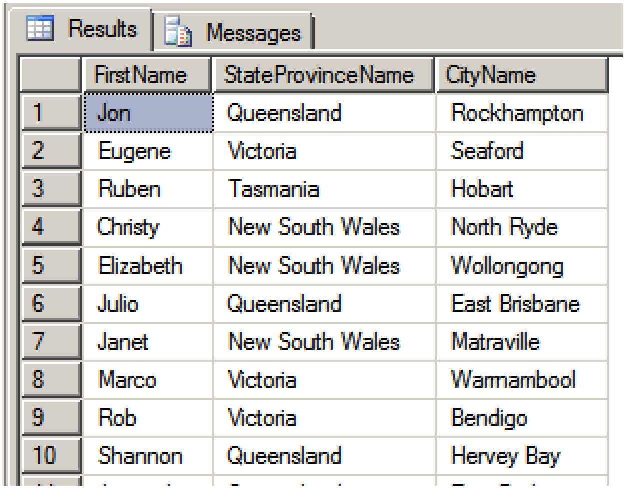
```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

An SQL join can merge multiple tables at a time

```
SELECT
        t1.FirstName,
        t2.StateProvinceName,
        t2.CityName
FROM dbo.DimCustomer AS t1
INNER JOIN dbo.DimGeography
AS t2 ON (t1.GeographyKey =
t2.GeographyKey)
```



	FirstName	StateProvinceName	CityName
1	Jon	Queensland	Rockhampton
2	Eugene	Victoria	Seaford
3	Ruben	Tasmania	Hobart
4	Christy	New South Wales	North Ryde
5	Elizabeth	New South Wales	Wollongong
6	Julio	Queensland	East Brisbane
7	Janet	New South Wales	Matraville
8	Marco	Victoria	Warrnambool
9	Rob	Victoria	Bendigo
10	Shannon	Queensland	Hervey Bay

- The advantage of using table alias (t1, t2 etc.) is that merging column can be of different names in different tables
- Joins can happen on multiple columns too

Further reading and practice

- Data warehousing concepts (Link)
- Interactive SQL Tutorial (Link)
- Beginning SQL Server 2008 for Developers (Chapters 11 and 12) (Link)
- SQL Server 2012: T-SQL Fundamentals (Link)
- SQL Server 2008: T-SQL Querying (Link)