



Module 5

Project lab & Quiz challenge



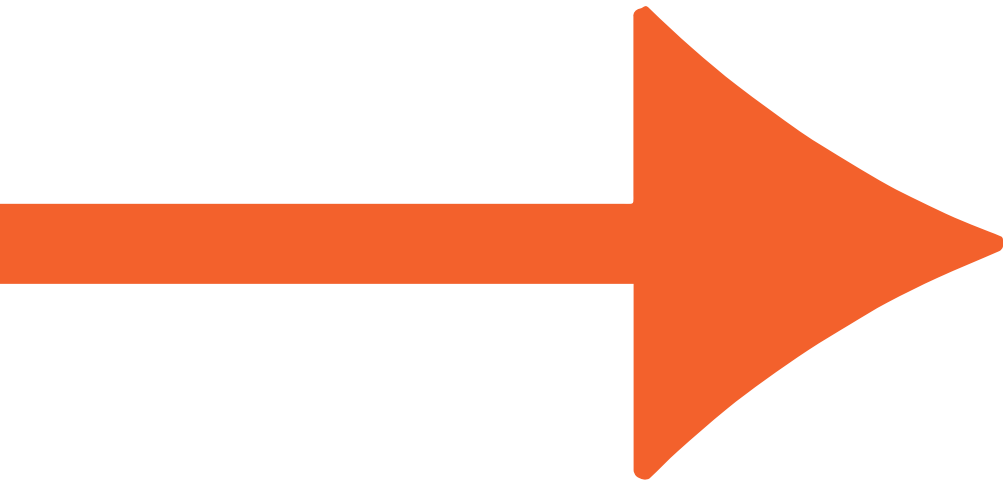


Project lab

- Outline

Use the knowledge of the SQL language from the past 4 days to create a database with a range of Tables, Views, Procedures & Functions.

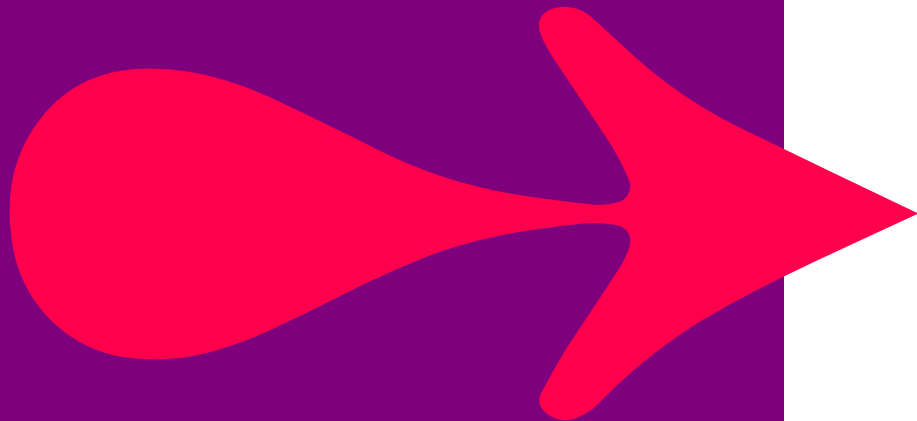
1. Come up with a sensible Normalized design for your database.
2. Create the database tables, data types and relationships for your Database.
3. Populate your database with some sample data.
4. Create a range of SQL queries for that sample data.
5. Write some examples of SQL Views, SQL Procedures, SQL Functions...for your database.





PROJECT LAB:

CREATE A DATABASE



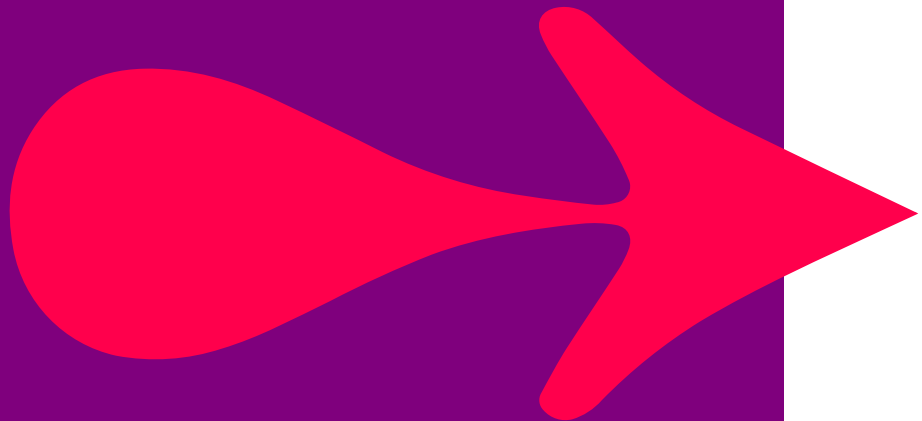
- **Exercise 1:** Based on the requirements, work out a normalized database design for the tables and columns, including the appropriate data types.
- **Exercise 2:** Using SQL, create the objects (Database, Tables, Relationships). Populate the tables with some sample data.
- **Exercise 3:** Write some SQL queries for your database.
- **Exercise 4:** Using SQL, create some Views, Procedures and Functions. Test them to ensure they work.
- Estimated Time: **4 hours**



PROJECT LAB:

EXERCISE 1

- **Exercise 1:** Based on the requirements, work out a normalized database design for the tables and columns, including the appropriate data types.
- Estimated Time: **60 mins incl. discussion**





Task 1 – Based on this requirement, plan what table and column names should be used. (About 15 mins – then discussion)

Your client, **Amazon Web Services**, has asked you to create a database for them to keep a track of their customers, which products their customers use on the Amazon website, what sales orders have been made by their customers and where those orders were placed (which AWS region). This data will only need to be stored in English. This is what they said:

We need a database to keep track of our customers, the AWS products our customers use, the orders those customer have mode and which AWS region they used when they submitted their order.

- *Each **customer** should have a first name, last name, birth date, mobile number, city and country.*
- *Each **product** should include a name, description, and web url.*
- *Each **region** needs a region name, region, unique country code.*
- *Finally, each **order** needs to include the date, customer, product, region and the amount of the order*

Recommendation: Using pen and paper, sketch out the table and column names. Leave space to the right for the data types and a few other columns we'll add in the next task.

Region data URL - <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.RegionsAndAvailabilityZones.html>



Task 1 – Recommended solution

Customers
CustomerID
FirstName
LastName
BirthDate
MobileNumber
City
Country

Products
ProductID
Name
Description
WebUrl

Regions
RegionID
Name
RegionCode
CountryCode

Orders
OrderID
CustomerID
ProductID
RegionID
OrderAmount
OrderDate

Notes: You may come up with slightly different table or column names but you format should be consistent and adopt a 'CamelCase' approach for the columns with no spacing.



Task 2 – Based on this requirement, plan the data type, nullability and a possible identity requirement of each column (About 15 mins – then discussion)

Your client, Amazon Web Services, has asked you to create a database for them to keep a track of their customers, which products their customers use on the Amazon website, what sales orders have been made by their customers and where those orders were placed (which AWS region). This data will only need to be stored in English. This is what they said:

We need a database to keep track of our customers, the AWS products our customers use, the orders those customer have mode and which AWS region they used when they submitted their order.

- Each **customer** should have a first name, last name, birth date, mobile number, city and country.
- Each **product** should include a name, description, and web url.
- Each **region** needs a region name, region, unique country code.
- Finally, each **order** needs to include the date, customer, product, region and the amount of the order

Recommendation: Using pen and paper, add the relevant data types, nullability and possible identity columns references.

Region data URL - <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.RegionsAndAvailabilityZones.html>



Task 2 – Recommended solution for Customers table

Customers	Date Type	Nullable	Identity
CustomerID	INT	No	Yes (1,1)
FirstName	VARCHAR (100)	No	No
LastName	VARCHAR (100)	Yes	No
BirthDate	DATE	No	No
MobileNumber	VARCHAR (20)	Yes	No
City	VARCHAR (100)	No	No
Country	VARCHAR (100)	No	No

Notes: You may come up with different values for the VARCHAR lengths. But do some google searches. There are some very long city and country names. And you may have some different ideas about the nullable values.



Task 2 – Recommended solution for Products table

Products	Date Type	Nullable	Identity
ProductID	INT	No	Yes (1,1)
Name	VARCHAR (100)	No	No
Description	VARCHAR (2000)	No	No
WebUrl	VARCHAR (300)	No	No

Notes: You may come up with different values for the VARCHAR lengths here. You would normally ask the client how long they want the product descriptions to be.



Task 2 – Recommended solution for Regions table

Regions	Date Type	Nullable	Identity
RegionID	SMALLINT	No	Yes (1,1)
Name	VARCHAR (40)	No	No
RegionCode	VARCHAR (30)	No	No
CountryCode	CHAR (3)	No	No

Notes:

Region codes: <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.RegionsAndAvailabilityZones.html>

Country codes: <https://www.iban.com/country-codes>



Task 2 – Recommended solution for Orders table

Orders	Date Type	Nullable	Identity
OrderID	INT	No	Yes (1,1)
CustomerID	INT	No	No
ProductID	INT	No	No
RegionID	SMALLINT	No	No
OrderAmount	MONEY	No	No
OrderDate	DATE	No	No

Notes: Depending on the client needs, a simple date data type may be acceptable or a more precise datetime data type might be required.



Task 3 – Based on these tables, work out the references for the Primary to Foreign Key relationships (About 5 mins – then discussion)

Customers
CustomerID
FirstName
LastName
BirthDate
MobileNumber
City
Country

Products
ProductID
Name
Description
WebUrl

Regions
RegionID
Name
RegionCode
CountryCode

Orders
OrderID
CustomerID
ProductID
RegionID
OrderAmount
OrderDate



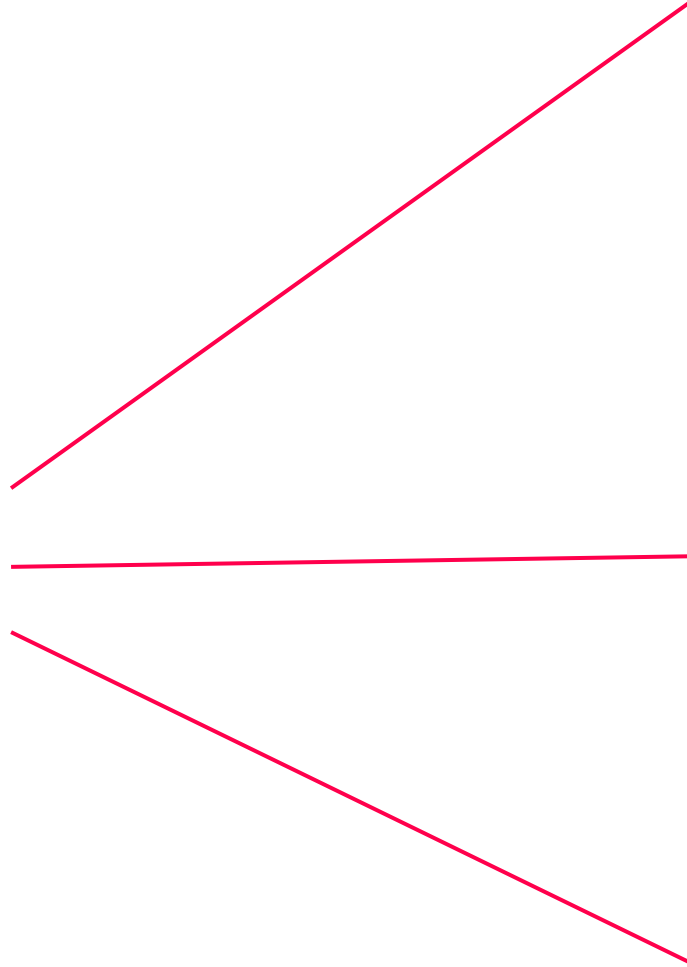
Task 3 – Recommended Solution

Orders
OrderID (PK)
CustomerID (FK)
ProductID (FK)
RegionID (FK)
OrderAmount
OrderDate

Customers
CustomerID (PK)
FirstName
LastName
BirthDate
MobileNumber
City
Country

Products
ProductID (PK)
Name
Description
WebUrl

Regions
RegionID (PK)
Name
RegionCode
CountryCode





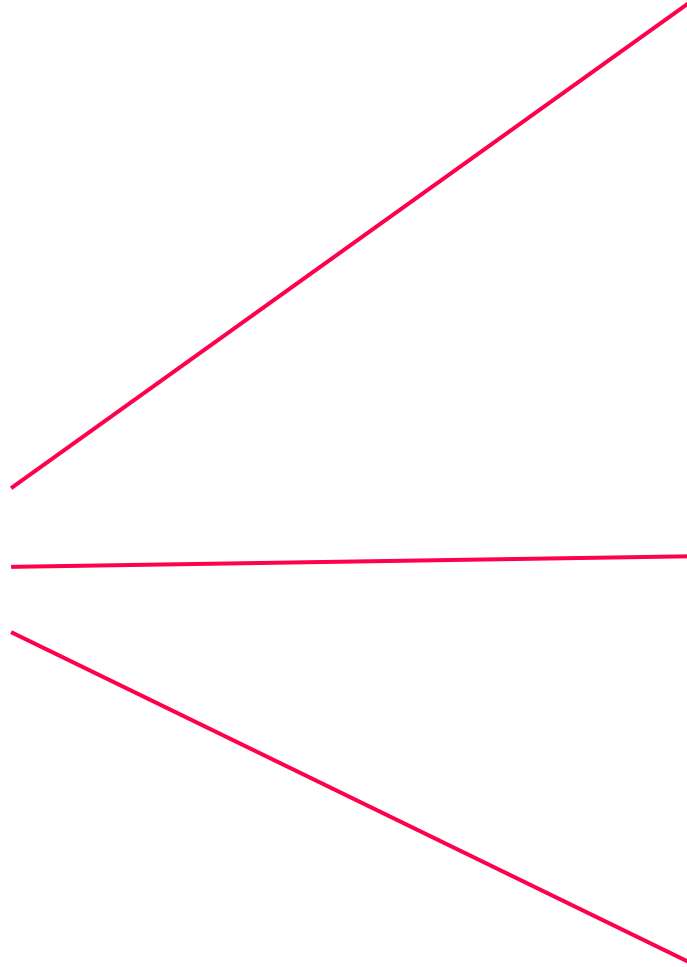
Task 3 – Recommended Solution

Orders
OrderID (PK)
CustomerID (FK)
ProductID (FK)
RegionID (FK)
OrderAmount
OrderDate

Customers
CustomerID (PK)
FirstName
LastName
BirthDate
MobileNumber
City
Country

Products
ProductID (PK)
Name
Description
WebUrl

Regions
RegionID (PK)
Name
RegionCode
CountryCode





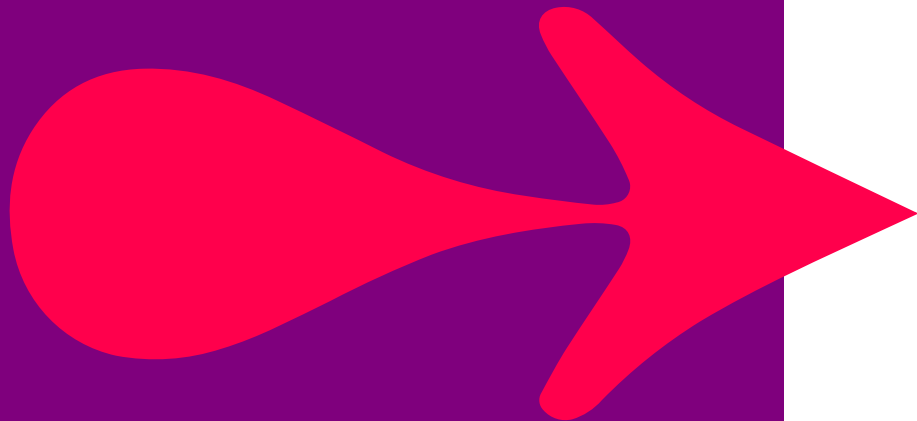
PROJECT LAB:

EXERCISE 2

Exercise 2: Based on the Database and Table plan, create these objects using SQL:

- Database
- Tables
- Relationships
- Populate the tables with some sample data

Estimated Time: **90 mins incl. any guidance and discussion**





Task 1 – What Database name will you use?

Write the code to create it. (Time – about 2mins)



Orders
OrderID (PK)
CustomerID (FK)
ProductID (FK)
RegionID (FK)
OrderAmount
OrderDate

Customers
CustomerID (PK)
FirstName
LastName
BirthDate
MobileNumber
City
Country

Products
ProductID (PK)
Name
Description
WebUrl

Regions
RegionID (PK)
Name
RegionCode
CountryCode



Task 2 – Create the Tables

Start by writing the SQL code to create the Customers Table in this slide. Does the SSMS console have a tool to help you get started? (Time – about 20mins)

Customers	Date Type	Nullable	Identity
CustomerID	INT	No	Yes (1,1)
FirstName	VARCHAR (100)	No	No
LastName	VARCHAR (100)	Yes	No
BirthDate	DATE	No	No
MobileNumber	VARCHAR (20)	Yes	No
City	VARCHAR (100)	No	No
Country	VARCHAR (100)	No	No



Task 2 – Create the Tables

Write the SQL code for the Products Table (Time – about 10mins)

Products	Date Type	Nullable	Identity
ProductID	INT	No	Yes (1,1)
Name	VARCHAR (100)	No	No
Description	VARCHAR (2000)	No	No
WebUrl	VARCHAR (300)	No	No



Task 2 – Create the Tables

Write the SQL code for the Regions Table. (Time – about 10mins)

Regions	Date Type	Nullable	Identity
RegionID	SMALLINT	No	Yes (1,1)
Name	VARCHAR (40)	No	No
RegionCode	VARCHAR (30)	No	No
CountryCode	CHAR (3)	No	No



Task 2 – Create the Tables

Write the SQL code for the Orders Table. (Time – about 15mins)

Orders	Date Type	Nullable	Identity
OrderID	INT	No	Yes (1,1)
CustomerID	INT	No	No
ProductID	INT	No	No
RegionID	SMALLINT	No	No
OrderAmount	MONEY	No	No
OrderDate	DATE	No	No



Task 3 – Create the Primary to Foreign Key relationships

Write the code to create them
(Time – about 15mins)

Orders
OrderID (PK)
CustomerID (FK)
ProductID (FK)
RegionID (FK)
OrderAmount
OrderDate

Customers
CustomerID (PK)
FirstName
LastName
BirthDate
MobileNumber
City
Country

Products
ProductID (PK)
Name
Description
WebUrl

Regions
RegionID (PK)
Name
RegionCode
CountryCode

Hint: Find a Foreign Key in an existing Orders Table.

Script it out to a new Query window as an example to help you.



Task 4 – Populate the Tables with some sample data. Try inserting just a few records into the Customers Table.

Once you've worked out the code for that, your Instructor will provide a script for you to re-create the database up to this same point with some populated data. To use it, you may need to delete or rename your database (if it's named **AWSCustomers**)

(About 20mins – then discussion)

Customers
CustomerID
FirstName
LastName
BirthDate
MobileNumber
City
Country

Products
ProductID
Name
Description
WebUrl

Regions
RegionID
Name
RegionCode
CountryCode

Orders
OrderID
CustomerID
ProductID
RegionID
OrderAmount
OrderDate



PROJECT LAB:

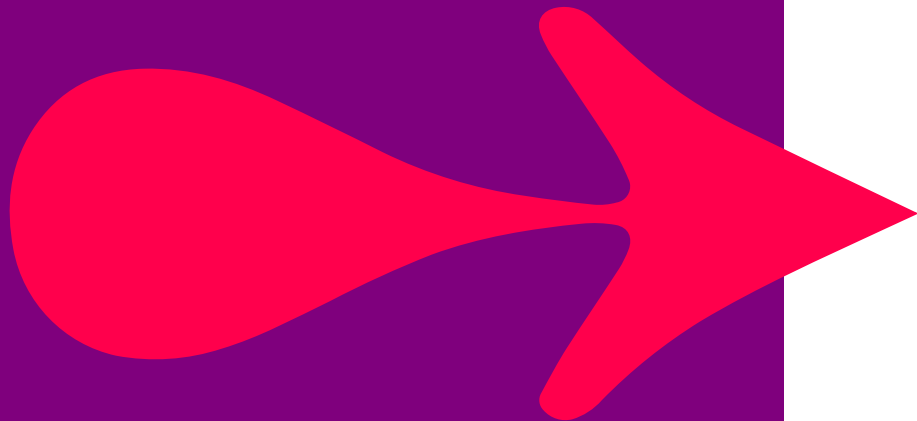
EXERCISE 3

Exercise 3: Write some SQL queries for the database.

Do as many as you can handle before your brain starts complaining about mistreatment...or you hit the lunch break!

There's a super tough one at the end for anyone that fancies themselves a SQL Query Master!

Estimated Time: **As many as possible. Finish them after lunch if necessary,**





Using the AWSCustomers database, you need to write some queries for the Sales Manager.

1. Write a query that displays the customer details for customers where their last name begins with the letter 'J'. Order the results by the last name.

Customers
CustomerID
FirstName
LastName
BirthDate
MobileNumber
City
Country

Products
ProductID
Name
Description
WebUrl



Regions
RegionID
Name
RegionCode
CountryCode

Orders
OrderID
CustomerID
ProductID
RegionID
OrderAmount
OrderDate



1. Write a query that displays the customer details for customers where their last name begins with the letter 'J'. Order the results by the last name.

```
SELECT *  
FROM dbo.Customers  
WHERE LastName LIKE 'J%'  
ORDER BY LastName
```

<div><div> Results</div><div> Messages</div></div>							
	CustomerID	FirstName	LastName	BirthDate	MobileNumber	City	Country
1	3	Richard	James	1975-07-22	07987862321	Paris	France
2	2	Sue	Jones	1980-10-03	07987654321	Wellington	New Zealand



2. Write a query that displays the order amounts for product id's 2 or 4 but only where the order amount is greater than 1000. Display the Order amount as 'Total Sales' and order the results from high to low.

Customers
CustomerID
FirstName
LastName
BirthDate
MobileNumber
City
Country

Products
ProductID
Name
Description
WebUrl

Regions
RegionID
Name
RegionCode
CountryCode

Orders
OrderID
CustomerID
ProductID
RegionID
OrderAmount
OrderDate



2. Write a query that displays the order amounts for product id's 2 or 4 but only where the order amount is greater than 1000. Display the Order amount as 'Total Sales' and order the results from high to low.

```
SELECT ProductID, OrderAmount AS 'Total Sales'  
FROM dbo.Orders  
WHERE ProductID IN (2, 4) AND OrderAmount > 1000  
ORDER BY 'Total Sales' DESC
```

Results Messages		
	ProductID	Total Sales
1	2	3290.20
2	2	2590.20
3	4	1600.10
4	4	1450.30



3. Write a query that displays the orders for all customers. The Manager needs to see their first name, last name, country, order amount and the date of the order.

Customers
CustomerID
FirstName
LastName
BirthDate
MobileNumber
City
Country

Products
ProductID
Name
Description
WebUrl

Regions
RegionID
Name
RegionCode
CountryCode

Orders
OrderID
CustomerID
ProductID
RegionID
OrderAmount
OrderDate



3. Write a query that displays the orders for all customers. The Manager needs to see their first name, last name, country, order amount and the date of the order.

```
SELECT c.FirstName, c.LastName, c.Country, o.OrderAmount, o.OrderDate
FROM dbo.Customers c
JOIN dbo.Orders o
ON c.CustomerID = o.CustomerID
```

Results		Messages			
	FirstName	LastName	Country	OrderAmount	OrderDate
1	Bob	Smith	United Kingdom	562.30	2023-06-10
2	Sue	Jones	New Zealand	1600.10	2023-07-03
3	Richard	James	France	850.90	2023-08-01
4	Mia	Hunter	Australia	3290.20	2023-08-01
5	Martin	Walsh	South Africa	562.30	2023-08-02
6	Bob	Smith	United Kingdom	700.30	2023-07-10
7	Sue	Jones	New Zealand	1450.30	2023-08-03
8	Richard	James	France	820.90	2023-09-01
9	Mia	Hunter	Australia	2590.20	2023-09-01
10	Martin	Walsh	South Africa	600.30	2023-09-02



4. Write a query that displays the number of orders that have been placed for each region. Order the results by the number of orders from high to low. Name the column 'Order Total'

Customers
CustomerID
FirstName
LastName
BirthDate
MobileNumber
City
Country

Products
ProductID
Name
Description
WebUrl

Regions
RegionID
Name
RegionCode
CountryCode

Orders
OrderID
CustomerID
ProductID
RegionID
OrderAmount
OrderDate



4. Write a query that displays the number of orders that have been placed for each region. Order the results by the number of orders from high to low. Name the column 'Order Total'

```
SELECT r.Name, COUNT(o.OrderID) AS 'Order Total'  
FROM dbo.Regions r  
JOIN dbo.Orders o  
ON r.RegionID = o.RegionID  
GROUP BY r.Name  
ORDER BY 'Order Total' DESC
```

Results Messages		
	Name	Order Total
1	Asia Pacific (Sydney)	4
2	Europe (London)	4
3	Africa (Cape Town)	2



5. Write a query that displays the orders for all customers. The Manager needs to see their first name, mobile number, country, product name, order amount and the date of the order. Order the results by the order amount from high to low.

Customers
CustomerID
FirstName
LastName
BirthDate
MobileNumber
City
Country

Products
ProductID
Name
Description
WebUrl

Regions
RegionID
Name
RegionCode
CountryCode

Orders
OrderID
CustomerID
ProductID
RegionID
OrderAmount
OrderDate



5. Write a query that displays the orders for all customers. The Manager needs to see their first name, mobile number, country, product name, order amount and the date of the order. Order the results by the order amount from high to low.

```
SELECT c.FirstName, c.MobileNumber, c.Country, p.Name, o.OrderAmount, o.OrderDate
FROM dbo.Customers c
JOIN dbo.Orders o
ON o.CustomerID = c.CustomerID
JOIN dbo.Products p
ON o.ProductID = p.ProductID
ORDER BY OrderAmount DESC
```

Results		Messages				
	FirstName	MobileNumber	Country	Name	OrderAmount	OrderDate
1	Mia	07987862161	Australia	Amazon Aurora	3290.20	2023-08-01
2	Mia	07987862161	Australia	Amazon Aurora	2590.20	2023-09-01
3	Sue	07987654321	New Zealand	Amazon VPC	1600.10	2023-07-03
4	Sue	07987654321	New Zealand	Amazon VPC	1450.30	2023-08-03
5	Richard	07987862321	France	Amazon Aurora	850.90	2023-08-01
6	Richard	07987862321	France	Amazon Aurora	820.90	2023-09-01
7	Bob	07123456789	United Kingdom	Amazon Aurora	700.30	2023-07-10
8	Martin	07974922161	South Africa	Amazon Aurora	600.30	2023-09-02
9	Bob	07123456789	United Kingdom	Amazon Aurora	562.30	2023-06-10
10	Martin	07974922161	South Africa	Amazon Aurora	562.30	2023-08-02



You are a SQL Query Master if you get this one right!

Write a query that displays the total order amount for each customer. The Manager needs the results grouped by the customer's mobile phone number.

The results should display the customer first name, last name, country, mobile phone, and orders as a total but only where the order total is greater than 1500. Alias the Order Total column appropriately and from high to low.

Customers
CustomerID
FirstName
LastName
BirthDate
MobileNumber
City
Country

Products
ProductID
Name
Description
WebUrl

Regions
RegionID
Name
RegionCode
CountryCode

Orders
OrderID
CustomerID
ProductID
RegionID
OrderAmount
OrderDate



Write a query that displays the total order amount for each customer. The Manager needs the results grouped by the customer's mobile phone number.

The results should display the customer first name, last name, country, mobile phone, and orders as a total.

```
SELECT c.FirstName, c.LastName, c.Country, c.MobileNumber, SUM(o.OrderAmount)
AS 'Sales Total'
FROM dbo.Customers c
JOIN dbo.Orders o
ON c.CustomerID = o.CustomerID
GROUP BY c.FirstName, c.LastName, c.Country, c.MobileNumber
HAVING SUM(o.OrderAmount) > 1500
ORDER BY 'Sales Total' DESC
```

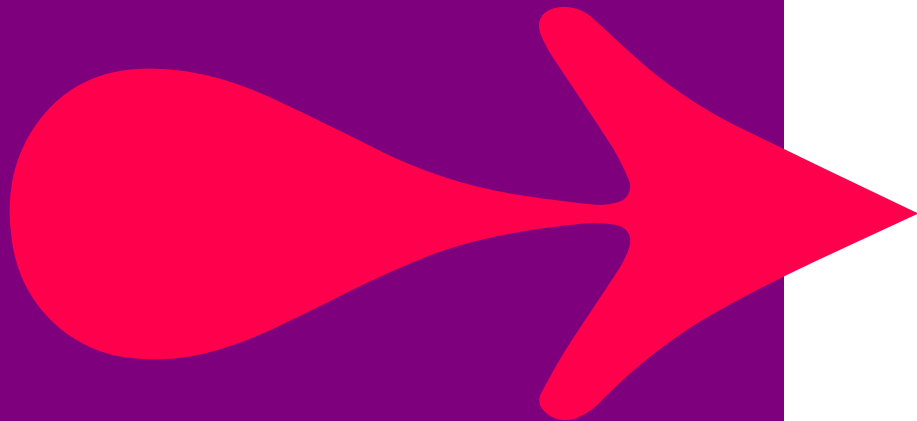
Results Messages					
	FirstName	LastName	Country	MobileNumber	Sales Total
1	Mia	Hunter	Australia	07987862161	5880.40
2	Sue	Jones	New Zealand	07987654321	3050.40
3	Richard	James	France	07987862321	1671.80



PROJECT LAB:

EXERCISE 4

- **Exercise 4:** Using SQL, create some Views, Procedures and Functions. Test them to ensure they work..
- Estimated Time: **Complete as many as possible by 2:30pm.**





Create a View

Write a View for one of the SELECT queries in the previous exercise.

Customers
CustomerID
FirstName
LastName
BirthDate
MobileNumber
City
Country

Products
ProductID
Name
Description
WebUrl

Regions
RegionID
Name
RegionCode
CountryCode

Orders
OrderID
CustomerID
ProductID
RegionID
OrderAmount
OrderDate



Create a View

Write a View for one of the SELECT queries in the previous exercise.

```
CREATE VIEW dbo.SalesOrdersCustomer
AS
SELECT c.FirstName, c.LastName, c.Country, c.MobileNumber, SUM(o.OrderAmount)
AS 'Sales Total'
FROM dbo.Customers c
JOIN dbo.Orders o
ON c.CustomerID = o.CustomerID
GROUP BY c.FirstName, c.LastName, c.Country, c.MobileNumber
HAVING SUM(o.OrderAmount) > 1500
```



Create a Procedure

Write a Procedure that displays the Orders for a Customer based on an input parameter of the Country.

Customers
CustomerID
FirstName
LastName
BirthDate
MobileNumber
City
Country

Products
ProductID
Name
Description
WebUrl

Regions
RegionID
Name
RegionCode
CountryCode

Orders
OrderID
CustomerID
ProductID
RegionID
OrderAmount
OrderDate



Create a Procedure

Write a Procedure that displays the Orders for a Customer based on an input parameter of the Country.

```
CREATE PROCEDURE dbo.SalesOrdersCountry
@Country varchar(100)
AS
SELECT c.FirstName, c.LastName, c.Country, o.OrderAmount, o.OrderDate
FROM dbo.Customers c
JOIN dbo.Orders o
ON c.CustomerID = o.CustomerID
WHERE c.Country = @Country
```

```
Exec dbo.SalesOrdersCountry @Country = 'United Kingdom'
```

Results		Messages			
	FirstName	LastName	Country	OrderAmount	OrderDate
1	Bob	Smith	United Kingdom	562.30	2023-06-10
2	Bob	Smith	United Kingdom	700.30	2023-07-10