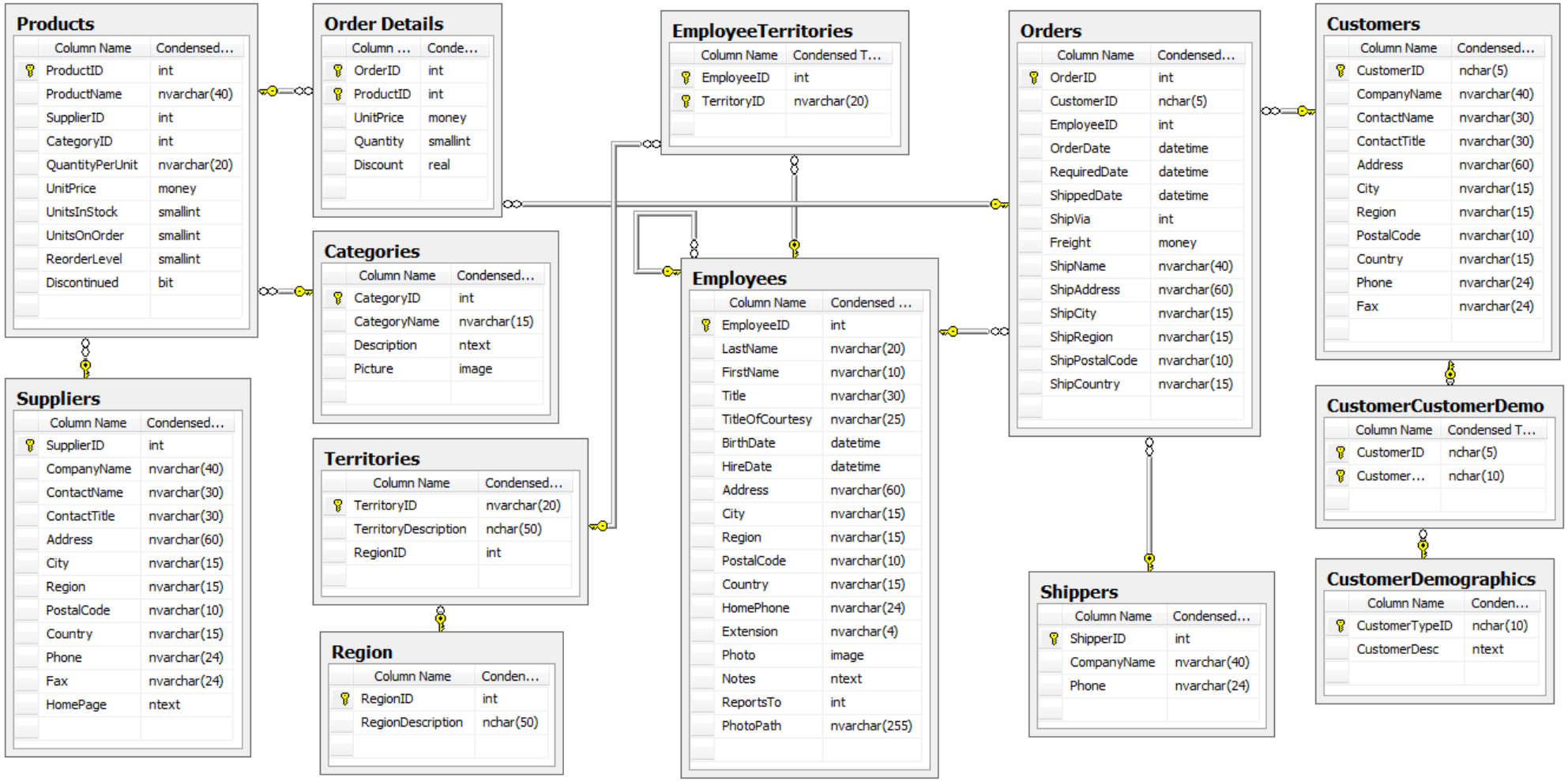
Introduction to Operating Systems and SQL for Data Science

Practice 9 - SQL

Northwind DB

- We will work with Nordwind DB for this recitation.



Reichman University

SQL – self learning

You are more than welcome to visit w3school
for further explanations and detailed documentation.

- http://www.w3schools.com/sql/default.asp

Reichman University

SQL Query Structure

SELECT <list_of_fields> ⇔ π<list of fields>
FROM <table_name>
WHERE <search_condition> ⇔ σ<search condition>

ORDER BY <list_of_fields>

Reichman University

SQL Query Structure – bit complex

SELECT <list_of_fields> ⇔ π<list of fields>
FROM <table_name>
WHERE <search_condition> ⇔ σ<search condition>

GROUP BY <list_of_fields>

HAVING <group_by_condition>

ORDER BY <list_of_fields>

Reichman University

SQL – Select .. From .. Where Example

```
SELECT *

FROM [Products]

WHERE ([UnitPrice] > 20 AND [UnitsOnOrder] <> 0) OR    ([ReorderLevel] = 10 AND
NOT [ProductID] = 3) OR ([UnitPrice] BETWEEN 8 AND 30)
```

Reichman University

SQL – Select .. From .. Where

Which products the following query selects?

```
SELECT *
FROM [Products]
WHERE [UnitPrice] IN (10,15,20)
```

Reichman University

SQL – Select .. From .. Where

Which products the following query selects?

SELECT *

FROM [Products]

WHERE [ProductName] **IS NOT Null**

Reichman University

SQL – “Like” conditions

Select ProductId and ProductName where product name contains “Alice”:

Reichman University

SQL – “Like” conditions

Select ProductId and ProductName where product name contains “Alice”:

SELECT [ProductID],[ProductName]

FROM [Products]

WHERE [ProductName] **LIKE** '%Alice%'

Reichman University

SQL – “Like” conditions

Select ProductId and ProductName where product name starts with “CH”:

Reichman University

SQL – “Like” conditions

Select ProductId and ProductName where product name starts with “CH”:

SELECT [ProductID],[ProductName]

FROM [Products]

WHERE [ProductName] **LIKE** 'CH%‘

Reichman University

SQL – “Like” conditions

Select ProductId and ProductName where product name starts with “CH” and contains 4 chars:

Reichman University

SQL – “Like” conditions

Select ProductId and ProductName where product name starts with “CH” and contains 4 chars:

SELECT [ProductID],[ProductName]

FROM [Products]

WHERE [ProductName] **LIKE** ‘CH__‘

Reichman University

SQL – Sorting result

Select CustomerID (sorted) from Orders table:

SELECT DISTINCT [CustomerID]

FROM [Orders]

ORDER BY [CustomerID], [field2], [field3] **DESC**

- **Distinct will avoid duplications**
- **Distinct should appear only once**
- **ASC – ascending DESC – descending**
- **We can sort by several fields (sort order is by fields order)**

Reichman University

SQL – Join

SELECT [field1],[fileld2],[field3],…

FROM [table1] A **JOIN** [table2] B

ON [table1].[field1] = [table2].[field2]

JOIN [table3] C

ON [table2].[field3] = [table3].[field4]

WHERE condition

Reichman University

SQL – Join

Another option:

SELECT [field1],[fileld2],[field3],…

FROM [table1] A,[table2] B, [table3] C

WHERE [table1].[field1] = [table2].[field2] **AND**

[table2].[field3] = [table3].[field4]

Reichman University

SQL – Join example with same table

Select product names of products that sells in at least 3 different colors :

```
SELECT DISTINCT p1.pname
FROM product AS p1, product AS p2, product AS p3
WHERE p1.pname=p2.pname AND p2.pname=p3.pname AND
  p1.color<>p2.color AND p1.color<>p3.color AND
  P2.color<>p3.color;
```

After renaming a table, we should approach it in its new name.

Reichman University

SQL – Join example with multiple tables

Select product ids names and suppliers of products from category 3 :

Reichman University

SQL – Join example with multiple tables

Select product ids names and suppliers of products from category 3 :

```
SELECT [ProductID],[ProductName],[CompanyName]
FROM [Products] JOIN [Suppliers]
ON [Products].[SupplierID] = [Suppliers].[SupplierID]
WHERE [CategoryID] = 3
```

Reichman University

SQL – Join example with multiple tables

Select product ids names and suppliers of products from category 3 :

```
SELECT [ProductID],[ProductName],[CompanyName]
FROM [Products],[Suppliers]
WHERE [Products].[SupplierID] = [Suppliers].[SupplierID]
        AND [CategoryID] = 3
```

Reichman University

SQL – computing new fields

Select for each product its total price in the order
(there might be Quantity >1)

```
SELECT [OrderID], [ProductID], [UnitPrice], [Quantity]
        ,[UnitPrice]*[Quantity] AS [Price]

FROM [Order Details]
```

- We defined new field (Price) for the total price in the order

Reichman University

SQL – computing new fields

```
SELECT [OrderID], [ProductID], [UnitPrice], [Quantity]
,[UnitPrice]*[Quantity] AS [Price]

FROM [Order Details]

Where [Price] > 100
```

We cannot use Price in the Where condition since Where is evaluated before the Select Action therefore not aware of the new field Price

Reichman University

The right way to do so…

Select for each product in Orders the total price of the product if total price is larger than 100, sort result by ProductId asc and by total price desc

SELECT [OrderID],[ProductID],[UnitPrice],[Quantity],[UnitPrice]*[Quantity] **AS** [Price]

FROM [Order Details]

WHERE [UnitPrice]*[Quantity] > 100

ORDER BY [ProductID], [UnitPrice]*[Quantity] **DESC**

Reichman University

SQL – Aggregation: max

Select maximum product price:

Reichman University

SQL – Aggregation: max

Select maximum product price:

SELECT MAX(UnitPrice) **AS** 'MaxPrice'

FROM [Products]

Reichman University

SQL – Aggregation: average

Select average price of product supplied by supplier 2:

Reichman University

SQL – Aggregation: average

Select average price of product supplied by supplier 2:

SELECT AVG([UnitPrice])

FROM [Products]

WHERE [SupplierID] = 2

Reichman University

SQL – Aggregation: sum

Select sum of prices of product supplied by supplier 2:

Reichman University

SQL – Aggregation: sum

Select sum of prices of product supplied by supplier 2:

SELECT SUM([UnitPrice])

FROM [Products]

WHERE [SupplierID] = 2

Reichman University

SQL – Aggregation: count

Select number of product supplied by supplier 2:

Reichman University

SQL – Aggregation: count

Select number of product supplied by supplier 2:

SELECT COUNT([ProductID]) **AS** 'Supplier2Products'

FROM [Products]

WHERE [SupplierID] = 2

Count(*) will return how many lines are in the result

Reichman University

SQL – Aggregation: count

Select number of suppliers that supply any product:

Reichman University

SQL – Aggregation: count

Select number of suppliers that supply any product:

SELECT COUNT(**DISTINCT** [SupplierID]) AS 'Suppliers'

FROM [Products]

Reichman University

SQL – Aggregation: Group by

Select for each supplier its ID and number of products it supplies:

Reichman University

SQL – Aggregation: Group by

Select for each supplier its ID and number of products it supplies:

SELECT [SupplierID],**COUNT**([ProductID]) AS 'Supplier2Products'

FROM [Products]

GROUP BY [SupplierID]

Reichman University

SQL – Aggregation: Group by

Select for each supplier its ID and number of products it supplies for suppliers with supplier id less than 15:

Reichman University

SQL – Aggregation: Group by

Select for each supplier its ID and number of products it supplies for suppliers with supplier id less than 15:

```
SELECT [SupplierID],COUNT([ProductID]) AS 'SuppliersProducts'

FROM [Products]

WHERE [SupplierID] < 15

GROUP BY [SupplierID]
```

Reichman University

SQL – Aggregation: Group by

Select for each supplier its ID and max product price is provides for suppliers with supplier id less than 15:

Reichman University

SQL – Aggregation: Group by

Select for each supplier its ID and max product price is provides for suppliers with supplier id less than 15:

SELECT [SupplierID], **MAX**([UnitPrice]) **AS** 'SupplierCostlyProductPrice'

FROM [Products]

WHERE [SupplierID] < 15

GROUP BY [SupplierID]

Reichman University

SQL – Aggregation: Group by + Having

Select for each supplier its ID and number of products it supplies for suppliers who provides more than 3 products:

Reichman University

SQL – Aggregation: Group by + Having

Select for each supplier its ID and number of products it supplies for suppliers who provides more than 3 products:

SELECT [SupplierID],**COUNT**([ProductID]) **AS** 'SuppliersProducts'

FROM [Products]

GROUP BY [SupplierID]

HAVING COUNT([ProductID]) > 3

Reichman University

SQL – Aggregation

What does the following query returns?

SELECT *

FROM [Products]

WHERE MAX([UnitPrice]) < 20

Reichman University

SQL – Aggregation

What does the following query returns?

SELECT *

FROM [Products]

WHERE MAX([UnitPrice]) < 20

- **This query is not valid since we can't use aggregation functions inside Where clause.**

Reichman University

SQL – Aggregation

Find for each order its total cost:

SQL – Aggregation

Find for each order its total cost:

SELECT [OrderID], **SUM** ([UnitPrice]*[Quantity]) **AS** [Price]

FROM [Order Details]

GROUP BY [OrderID]

Reichman University

SQL – Nested queries

Find product id and name of the best seller product:

Reichman University

SQL – Nested queries

Find product id and name of the best seller product:

```
SELECT [Products].[ProductID], [ProductName]
FROM [Order Details], [Products]
WHERE [Order Details].[ProductID] = [Products].[ProductID]
GROUP BY [Products].[ProductID],[ProductName]
HAVING COUNT(*) >= ALL (SELECT COUNT([OrderID])
                        FROM [Order Details]
                        GROUP BY [ProductID])
```

Reichman University

SQL – Nested queries: EXCEPT

Find product ids and names of the products with product id less or equal than 30:

SQL – Nested queries: EXCEPT

Find product ids and names of the products with product id less or equal than 30:

```
SELECT [ProductID],[ProductName]
FROM [Products]
EXCEPT (SELECT [ProductID],[ProductName]
        FROM [Products]
        WHERE [ProductID] > 30)
```

Reichman University

SQL – Nested queries: EXIST

Find product ids and names of the products that was ordered:

Reichman University

SQL – Nested queries: EXIST

Find product ids and names of the products that was ordered:

```
SELECT [ProductID], [ProductName]
FROM [Products] P
WHERE EXISTS (SELECT *
              FROM [Order Details]
              WHERE [ProductID] = P.[ProductID])
```

Reichman University

SQL – Nested queries: IN

Find product ids and names of the products that was ordered:

```
SELECT [ProductID], [ProductName]
FROM [Products]
WHERE [ProductID] IN (SELECT DISTINCT [ProductID]
                      FROM [Order Details])
```

Reichman University

SQL – INSERT

```
INSERT INTO [Table name]
        ([Field1],
        [Field2],
         …)
   VALUES
        (Value1,Value2,…)
```

```
INSERT INTO [Products]
        ([ProductName]
        ,[SupplierID]
        ,[CategoryID]
        ,[QuantityPerUnit]
        ,[UnitPrice]
        ,[UnitsInStock]
        ,[UnitsOnOrder]
        ,[ReorderLevel]
        ,[Discontinued])
   VALUES
        ('Banana','3','1','20','100','3','2','10','0')
```

Reichman University

SQL – UPDATE

Update UnitPrice to 12 for products with id 3:

UPDATE [Products]

SET [UnitPrice] = 12

WHERE [ProductID] = 3

Reichman University

SQL – DELETE

Delete products with product id greater than 50:

DELETE FROM [Products]

WHERE [ProductID] > 50

Reichman University

Question:

- We are given with the following tables schemas (Primary keys are underlined)

Products (**p_id**, p_name, p_description, category_id, cat_name, manuf_id, manuf_name)

Product_attributes (**p_id, attribute_name**, attribute_value)

Sites (**site_id**, site_name, site_url)

ProductPrice (**p_id, site_id, from_date**, to_date, price, product_site_url)

Reichman University

Question:

- We are given with the following tables schemas (Primary keys are underlined)
- **Find for product with p_id 18 the difference between its most expensive price in site to its lowest price in site (current prices. Current prices are the ones with to_date=NULL)**

Products (**p_id**, p_name, p_description, category_id, cat_name, manuf_id, manuf_name)

Product_attributes (**p_id, attribute_name**, attribute_value)

Sites (**site_id**, site_name, site_url)

ProductPrice (**p_id, site_id, from_date**, to_date, price, product_site_url)

Reichman University

Solution:

Products (**p_id**, p_name, p_description, category_id, cat_name, manuf_id, manuf_name)

Product_attributes (**p_id, attribute_name**, attribute_value)

Sites (**site_id**, site_name, site_url)

ProductPrice (**p_id, site_id, from_date**, to_date, price, product_site_url)

SELECT MAX(p1.price - p2.price)

FROM ProdcutPrice p1 **JOIN** ProdcutPrice p2

ON p1.p_id=p2.p_id **AND** p1.site_id<>p2.site_id

WHERE p1.pid=18 **AND** to_date is NULL

Reichman University