CZ2007: Databases

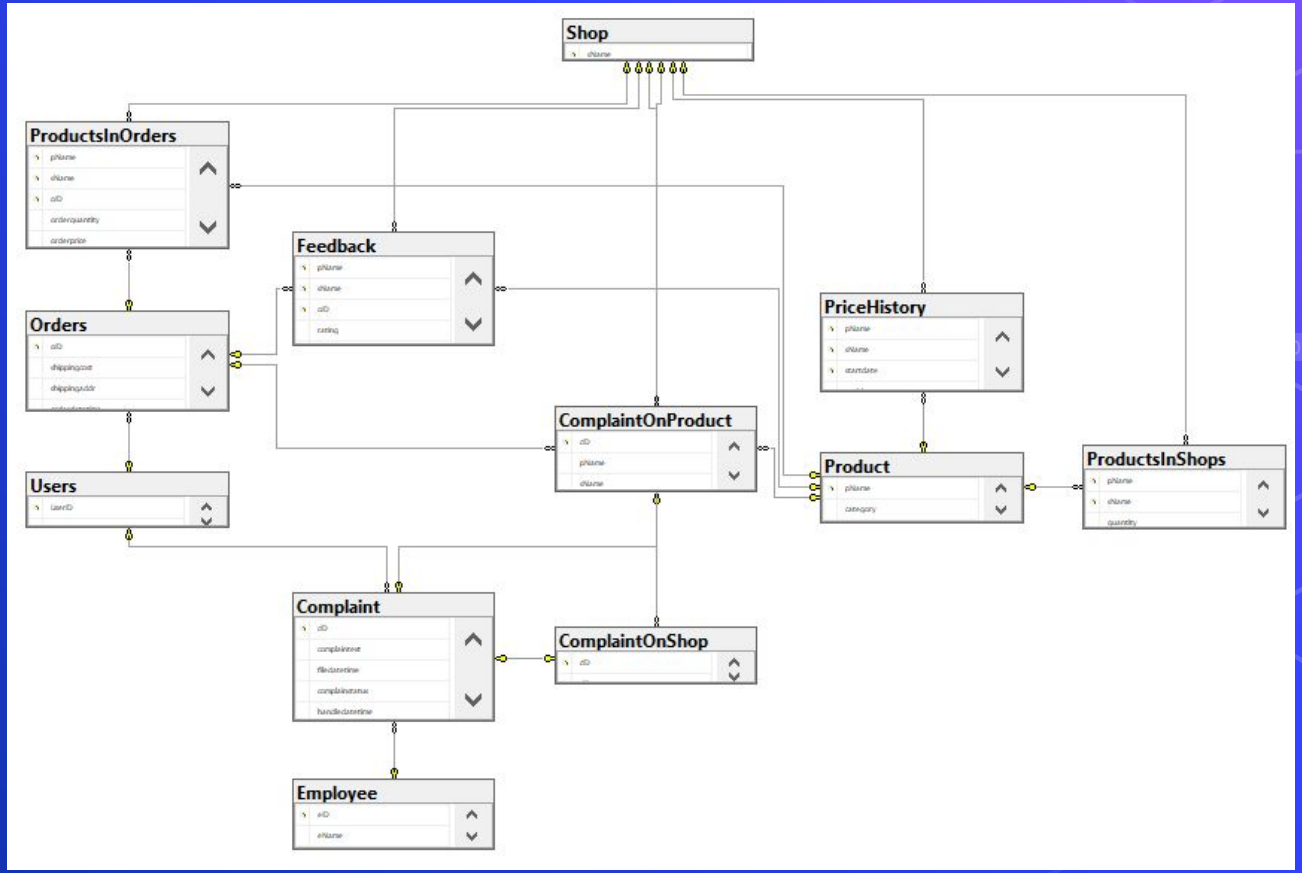
SSP5: Group 5

Entity Relationship Diagram

- Here you have a list of items
- And some text
- But remember not to overload your slides with content

Your audience will listen to you or read the content, but won’t do both.

Database Diagram



SQL Table Creation

```
CREATE TABLE ProductsInOrders(
    pName nvarchar(500) FOREIGN KEY REFERENCES Product(pName) ON DELETE CASCADE ON UPDATE CASCADE,
    sName varchar(100) FOREIGN KEY REFERENCES Shop(sName) ON DELETE CASCADE ON UPDATE CASCADE,
    oID int FOREIGN KEY REFERENCES Orders(oID) ON DELETE CASCADE,
    orderquantity int NOT NULL DEFAULT 0 CHECK(orderquantity>=0),
    orderprice float(24) NOT NULL DEFAULT 0.0 CHECK(orderprice>=0.0),
    deliverystatus varchar(50) NOT NULL DEFAULT 'being processed'
    Check(deliverystatus = 'being processed' OR deliverystatus = 'shipped' OR deliverystatus = 'delivered' OR deliverystatus = 'returned'),
    deliverydate datetime DEFAULT NULL,
    --The product in order either has DeliveryStatus = 'delivered' or 'returned' and a DeliveryDateTime.
--or DeliveryStatus 'being processed' or 'shipped' and DeliveryDateTime = NULL.
    CHECK((deliverystatus='delivered' AND deliverydate<>NULL)
    OR (deliverystatus='returned' AND deliverydate<>NULL)
    OR (deliverystatus='being processed' AND deliverydate=NULL)
    OR (deliverystatus='shipped' AND deliverydate=NULL)),
    PRIMARY KEY(pName, sName, oID),
);
```

Tables created along with primary keys, foreign keys, key constraints and tuple constraints

SQL Insert Dummy Data

```
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (1, N'aluckham0')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (2, N'cerbe1')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (3, N'lgouth2')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (4, N'ogoldsworthy3')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (5, N'thanmore4')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (6, N'rpawden5')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (7, N'aslight6')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (8, N'bilyuchyov7')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (9, N'bscarffe8')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (10, N'bstoffel9')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (11, N'uambrosonia')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (12, N'gduffieldb')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (13, N'sleiboldc')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (14, N'cleydend')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (15, N'erogete')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (16, N'cseczykf')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (17, N'jclaffeyg')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (18, N'emurrillh')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (19, N'mteodoroi')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (20, N'atutillj')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (21, N'eolerenshawk')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (22, N'zlarradl')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (23, N'wlerohanm')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (24, N'jjosowitzn')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (25, N'vmatkino')
INSERT [dbo].[Users] ([UserID], [uName]) VALUES (26, N'solenchikovp')
```

```
INSERT [dbo].[Product] ([pName], [maker], [category]) VALUES (N'Beans - Yellow', N'NULL', N'Grocery')
INSERT [dbo].[Product] ([pName], [maker], [category]) VALUES (N'Dell XPS Notebook', N'Dell', N'Electronic Gadgets')
INSERT [dbo].[Product] ([pName], [maker], [category]) VALUES (N'Doilies - 8, Paper', N'NULL', N'Grocery')
INSERT [dbo].[Product] ([pName], [maker], [category]) VALUES (N'Galaxy Note 10', N'Samsung', N'Electronic Gadgets')
INSERT [dbo].[Product] ([pName], [maker], [category]) VALUES (N'Galaxy S10', N'Samsung', N'Electronic Gadgets')
INSERT [dbo].[Product] ([pName], [maker], [category]) VALUES (N'Galaxy S10+', N'Samsung', N'Electronic Gadgets')
INSERT [dbo].[Product] ([pName], [maker], [category]) VALUES (N'Galaxy S20', N'Samsung', N'Electronic Gadgets')
INSERT [dbo].[Product] ([pName], [maker], [category]) VALUES (N'Galaxy S20+', N'Samsung', N'Electronic Gadgets')
INSERT [dbo].[Product] ([pName], [maker], [category]) VALUES (N'Galaxy S9', N'Samsung', N'Electronic Gadgets')
INSERT [dbo].[Product] ([pName], [maker], [category]) VALUES (N'Galaxy S9+', N'Samsung', N'Electronic Gadgets')
INSERT [dbo].[Product] ([pName], [maker], [category]) VALUES (N'Glass Cup', N'IKEA', N'Tableware')
```

SQL Triggers:

```
GO
CREATE TRIGGER NoUserUpdate ON Users
AFTER UPDATE
AS
IF UPDATE(UserID)
BEGIN
   ;THROW 51000, 'You can''t update the primary key UserID', 1;
END
GO
CREATE TRIGGER NoEmployeeUpdate ON Employee
AFTER UPDATE
AS
IF UPDATE(eID)
BEGIN
   ;THROW 51000, 'You can''t update the primary key employeeID', 1;
END
GO
CREATE TRIGGER NoOrderUpdate ON Orders
AFTER UPDATE
AS
IF UPDATE(oID)
BEGIN
   ;THROW 51000, 'You can''t update the primary key orderID', 1;
END
```

- Triggers to ensure that User ID, employee ID and order ID, cannot be changed

SQL Triggers:

- Ensure Complain status can only move up a stage at a time
  Eg: (being handled → addressed) is allowed but (pending → addressed) is not allowed

- Update Date & Time when the complaint has been addressed

```
GO
CREATE TRIGGER ComplainStatus
ON Complaint
AFTER UPDATE
NOT FOR REPLICATION
AS
BEGIN
    UPDATE  Complaint
    SET     complainstatus= CASE
                                WHEN d.eID = NULL AND i.eID IS NOT NULL
                                    THEN 'being handled'

                                WHEN d.complainstatus='being handled' AND i.complainstatus<>'addressed'
                                    THEN 'being handled'

                                WHEN d.complainstatus='pending' AND i.complainstatus<>'being handled'
                                    THEN 'pending'
                                ELSE
                                    i.complainstatus
                            END,
            handledatetime= CASE
                                WHEN d.complainstatus='pending' AND i.complainstatus='being handled'
                                    THEN GETDATE()
                                ELSE
                                    d.complainstatus
                            END
    FROM    Complaint o, inserted i, deleted d
    WHERE   o.cID=i.cID AND o.cID = d.cID

END
```

SQL Triggers:

- Make sure that the delivery status can only move up by a stage at a time (being processed → shipped)
- Update date & time when the item has been delivered

```
--Update DeliveryDateTime when DeliveryStatus changed.
--If DeliveryStatus changed to 'Delivered', then DeliveryDateTime=GETDATE()
--If DeliveryStatus changed to 'Pending', then DeliveryDateTime=NULL
--One trigger, one batch of statements
CREATE TRIGGER UpdateDelivery
ON ProductsInOrders
AFTER UPDATE
NOT FOR REPLICATION
AS
BEGIN
    UPDATE  ProductsInOrders
    --DeliveryDateTime will not be updated unless DeliveryStatus changes from 'shipped' to 'delivered'
    SET     deliverydate=   CASE
                                --If previous DeliveryStatus='shipped' and is changed to 'delivered', then DeliveryDateTime=GETDATE().
                                WHEN d.deliverystatus='shipped' AND i.deliverystatus='delivered'
                                    THEN GETDATE()
                                --DeliveryDateTime retains the old value
                                ELSE
                                    d.deliverydate
                            END
    --DeliveryStatus will not be updated unless if follows the sequence: 'being processed'->'shipped'->'delivered'->'returned'
            ,deliverystatus=    CASE
                                --If previous DeliveryStatus='being processed'. It can only be changed to 'shipped'.
                                WHEN d.deliverystatus='being processed' AND i.deliverystatus<>'shipped'
                                    THEN 'being processed'
                                --If previous DeliveryStatus='shipped'. It can only be changed to 'delivered'.
                                WHEN d.deliverystatus='shipped' AND i.deliverystatus<>'delivered'
                                    THEN 'shipped'
                                --If previous DeliveryStatus='delivered'. It can only be changed to 'returned'.
                                WHEN d.deliverystatus='delivered' AND i.deliverystatus<>'returned'
                                    THEN 'delivered' --DeliveryStatus retains updated value
                                WHEN d.deliverystatus='returned'
                                    THEN 'returned'

                                ELSE
                                    i.deliverystatus
                            END
    FROM    ProductsInOrders o, inserted i, deleted d
    --Get all the records that have just been updated,
    --and find the previous value (inserted gives the updated rows, and deleted gives the previous values for these rows)
    WHERE   o.SName=i.SName AND o.PName=i.PName AND o.oID=i.oID
    AND o.SName=d.SName AND o.PName=d.PName AND o.oID=d.oID;
END
```

Queries

Query 1

Find the average price of “iPhone X”s on Sharkee from 1 August 2020 to 31 August 2020.

Query 1:

```
---- Query 1 ----
-- Question: Find the average price of “iPhone Xs” on Sharkee from 1 August 2020 to 31 August 2020.

SELECT AVG(price) AS AvgPrice
FROM PriceHistory
WHERE pName = 'iPhone X'
AND ((startdate >= '2020.08.01 00:00:00' AND startdate < '2020.09.01 00:00:00')
    OR (enddate >= '2020.08.01 00:00:00' AND enddate < '2020.09.01 00:00:00'));
```

Output:

	AvgPrice
1	977.3333333333334

Query 2

Find products that received at least 100 ratings of “5” in August 2020, and order them by their average ratings.

Query 2(version-1):

```
-- create temporary table which stores product name with more than 100 ratings of "5"
SELECT pName INTO #Pdts
FROM Feedback
WHERE rating = 5 AND MONTH(feedbackDate) = 8 AND YEAR(feedbackDate) = 2020
GROUP BY pName
HAVING COUNT(rating)>=100;

-- printing the average ratings for these products
SELECT pName, ROUND(AVG(Cast(rating as Float)),2) AS AvgRatings
FROM Feedback
WHERE pName IN(SELECT * FROM Pdts) AND MONTH(feedbackDate) = 8 AND YEAR(feedbackDate) = 2020
GROUP BY pName
ORDER BY AVG(rating) DESC;
```

Output:

	pName	AvgRatings
1	Galaxy S9	4.95
2	iPhone X	4.84

Query 2(version-2):

```sql
-- Only extract those rows whereby the month of the feedback is August(08) and the year is 2020
WITH F0 AS ( --products that receive feedback in August 2020
SELECT *
FROM Feedback f
WHERE MONTH(feedbackDate) = 8 AND YEAR(feedbackDate) = 2020),

-- products that received ratings 5
F1 AS(
SELECT *
FROM F0
WHERE rating=5),

-- products with at least 100 ratings of 5
F2 AS(
SELECT pName, COUNT(rating) AS NumRatings5
FROM F1
GROUP BY pName
HAVING COUNT(rating)>=100)

-- First cast Rating to be a Float so we can get the decimal point values, then Average all the Rating values
-- Round the average rating to be 2 decimal point
SELECT F2.pName, ROUND(AVG(Cast(F0.rating as Float)),2) AS AvgRatings
FROM F2
JOIN F0 ON F2.pName=F0.pName
GROUP BY F2.pName
ORDER BY AvgRatings DESC;
```

Output:

	pName	AvgRatings
1	Galaxy S9	4.95
2	iPhone X	4.84

Query 3

For all products purchased in June 2020 that have been delivered, find the average time from the ordering date to the delivery date.

Query 3:

```
---- Query 3 ----
-- Question: For all products purchased in June 2020 that have been delivered, find the average time from the ordering date to the delivery date.

-- print average time of delivery in hours
SELECT CAST(AVG(DATEDIFF(second,orderdatetime, deliverydate)) AS FLOAT)/3600 AS AvgTimeOnDelivery
FROM Orders, ProductsInOrders
WHERE Orders.oID= ProductsInOrders.oID
AND orderdatetime >= '2020-06-01 00:00:01' AND orderdatetime <= '2020-06-30 23:59:59'
AND (deliverystatus = 'Delivered' OR deliverystatus = 'Returned');
```

Output:

	AvgTimeOnDelivery
1	116.3475

Query 4

Let us define the “latency” of an employee by the average that he/she takes to process a complaint. Find the employee with the smallest latency.

Query 4

```
---- Query 4 ----
-- Question: Let us define the "latency" of an employee by the average that he/she takes to process a complaint.
--Find the employee with the smallest latency.

SELECT eID, AVG(DATEDIFF(second, filedatetime, handledatetime)) AS AvgLatency INTO LatencyRecord
FROM Complaint
GROUP BY eID;

-- cross-check table
SELECT *
FROM LatencyRecord;

SELECT eID
FROM LatencyRecord
WHERE AvgLatency = (SELECT MIN(AvgLatency) FROM LatencyRecord);
```

Output:

	eID
1	4

Table for cross checking the answer

	eID	AvgLatency
1	1	2704800
2	2	2880000
3	3	2678400
4	4	2160000

Query 5

Produce a list that contains (i) all products made by Samsung, and (ii) for each of them, the number of shops on Sharkee that sell the product.

Query 5 (i):

```
---- Query 5 ----
-- Question: Produce a list that contains (i) all products made by Samsung, and
--(ii) for each of them, the number of shops on Sharkee that sell the product.

-- Part(i) --
SELECT pName
FROM Product
WHERE maker = 'Samsung';
```

Output:

	pName
1	Galaxy Note 10
2	Galaxy S10
3	Galaxy S10+
4	Galaxy S20
5	Galaxy S20+
6	Galaxy S9
7	Galaxy S9+
8	Samsung Galaxy Tab S5
9	Samsung LCD TV

Query 5 (ii):

```sql
-- Part(ii) -- version 1
SELECT Product.pName, COUNT(Sname) AS noOfShops
FROM ProductsInShops RIGHT JOIN Product ON Product.pName = ProductsInShops.pName
WHERE maker = 'Samsung'
GROUP BY Product.pName;

-- Part(ii) -- version 2
SELECT Product.pName AS Product, COUNT(ProductsInShops.PName) AS noOfShops
FROM Product
LEFT JOIN ProductsInShops
ON Product.pName = ProductsInShops.pName
WHERE maker = 'Samsung'
GROUP BY Product.pName;
```

Query 5:

Output:

	Product	noOfShops
1	Galaxy Note 10	0
2	Galaxy S10	1
3	Galaxy S10+	0
4	Galaxy S20	1
5	Galaxy S20+	1
6	Galaxy S9	1
7	Galaxy S9+	1
8	Samsung Galaxy Tab S5	1
9	Samsung LCD TV	1

Query 6

Find shops that made the most revenue in August 2020.

Query 6:

```
---- Query 6 ----
-- Quesiton: Find shops that made the most revenue in August 2020.

WITH A1 AS
(
 SELECT t2.sName, SUM(t2.orderprice*t2.orderquantity) AS revenue
 FROM Orders as t1

 -- Left join on common attribute OrderID of both tables
 LEFT JOIN ProductsInOrders AS t2
 ON t1.oID = t2.oID

 -- OrderDateTime should fall under 2020/08
 WHERE MONTH(t1.orderdatetime) = 8 AND YEAR(t1.orderdatetime) = 2020
 -- Group by Shop name with aggregate function SUM of all revenue(OrderPrice*OrderQuantity) by this shop
 GROUP BY sName
)


SELECT sName
FROM A1
WHERE revenue = (SELECT MAX(revenue) FROM A1);
```

Cross Verify the Total Revenue:

	sName	revenue
1	iStudio	154000
2	Samsung	110000
3	Tesco	462799
4	Walmart	590525

Output:

	sName
1	Walmart

Query 7

For users that made the most amount of complaints, find the most expensive products he/she has ever purchased.

Query 7:

```
---- Query 7 ----
-- Question: For users that made the most amount of complaints, find the most expensive products he/she has ever purchased.

-- Counts the total number of complaints each user has made
WITH A1 AS
(
 SELECT UserID, COUNT(UserID) as noOfComplaints
 FROM Complaint
 GROUP BY UserID
),

-- Select the users in A1 that has made the most complaints and their orderID
A2 AS
(
 SELECT t1.UserID, t2.oID
 FROM A1 as t1
 LEFT JOIN Orders as t2
 ON t1.UserID = t2.UserID
 WHERE noOfComplaints = (SELECT MAX(noOfComplaints) FROM A1)
),

-- Find all products that these users in A2 has ever purchased
A3 AS
(
 SELECT t1.UserID, t2.oID, t2.pName, t2.orderprice
 FROM A2 as t1
 LEFT JOIN ProductsInOrders as t2
 ON t1.oID = t2.oID

),

-- Find the most expensive product that each user in A3 has purchased
A4 AS
(
 SELECT UserID, MAX(OrderPrice) as maxProductPrice FROM A3
 GROUP BY UserID
)

-- Get the product name by matching UserID and the Product price
SELECT t1.UserID, t2.pName, t2.orderprice
FROM A4 as t1
LEFT JOIN A3 as t2
ON t1.UserID = t2.UserID AND t1.maxProductPrice = t2.OrderPrice;
```

Output:

	UserID	pName	orderprice
1	4	iPhone X	1400
2	87	Pureboost Shoes	179

Query 8

Find products that have never been purchased by some users, but are the top 5 most purchased products by other users in August 2020.

Query 8:

```
---- Query 8 ----
-- Question: Find products that have never been purchased by some users, but are the top 5 most purchased products by other users in August 2020.

-- Create a view with all products sold in August, group by PName, and find total quantity
GO
CREATE VIEW AllProducts AS
(SELECT pName, SUM(orderquantity) AS TotalQuantity
FROM ProductsInOrders PIO
JOIN Orders O ON PIO.oID =O.oID AND
(O.OrderDateTime >= '2020.08.01 00:00:00' AND O.OrderDateTime < '2020.09.01 00:00:00')
GROUP BY PName);

GO
-- View that excludes top product in August
CREATE VIEW NonTop1Products AS
(SELECT *
FROM AllProducts AP
WHERE AP.TotalQuantity <> (SELECT MAX(TotalQuantity)
                           FROM Allproducts AP2));

GO
-- View that excludes top 2 product in August
CREATE VIEW NonTop2Products AS
(SELECT *
FROM NonTop1Products NT
WHERE NT.TotalQuantity <> (SELECT MAX(TotalQuantity)
                           FROM NonTop1Products NT1));

GO
-- View that excludes top 3 product in August
CREATE VIEW NonTop3Products AS
(SELECT *
FROM NonTop2Products NT
WHERE NT.TotalQuantity <> (SELECT MAX(TotalQuantity)
                           FROM NonTop2Products NT1));

GO
-- View that excludes top 4 product in August
CREATE VIEW NonTop4Products AS
(SELECT *
FROM NonTop3Products NT
WHERE NT.TotalQuantity <> (SELECT MAX(TotalQuantity)
                           FROM NonTop3Products NT1));
```

Query 8 continued:

```
GO
-- View that excludes top 5 product in August
CREATE VIEW NonTop5Products AS
(SELECT *
FROM NonTop4Products NT
WHERE NT.TotalQuantity <> (SELECT MAX(TotalQuantity)
                           FROM NonTop4Products NT1));

GO
-- View that get top 5 product in August
-- Assume that there can be more than 5 products if products have the same order quantity.
CREATE VIEW TopProducts AS
(SELECT *
FROM AllProducts
EXCEPT
SELECT *
FROM NonTop5Products);

GO
-- View that gets the number of unique users
CREATE VIEW UserCount AS
SELECT COUNT(*) AS NumUniqueUsers
FROM Users;

GO
-- View that gets the number of unique purchases for each product
CREATE VIEW UniquePurchases AS
SELECT pName, Count(UserID) AS NumUniquePurchases
FROM (SELECT DISTINCT U.UserID, pName
      FROM Users U, Orders O ,ProductsInOrders PIO
      WHERE U.UserID=O.UserID AND O.oID=PIO.oID) AS UniquePurchase
GROUP BY pName;

GO
-- View that gets the products that are not bought by some users, but are top 5 products
SELECT DISTINCT TP.pName
FROM TopProducts TP, UserCount UC,UniquePurchases UP
WHERE TP.pName=UP.pName AND NumUniquePurchases < NumUniqueUsers;
```

Query 8 continued:

```
GO

-- additional commands to visualise views
-- All products
SELECT *
FROM AllProducts
ORDER BY TotalQuantity DESC;

-- Top 5 products
SELECT *
FROM TopProducts
ORDER BY TotalQuantity DESC;

-- Number of unique users
SELECT *
FROM UserCount;

-- Number of  unique purchases for each product
SELECT *
FROM UniquePurchases;

-- Number of  unique purchases for each product with number of unique users added
SELECT TP.pName, NumUniquePurchases, NumUniqueUsers
FROM TopProducts TP, UserCount UC,UniquePurchases UP
WHERE TP.pName=UP.pName;
```

Query 8 Results:

Output:

	pName
1	Huawei P40
2	JBL speaker
3	Men AIRism Crew Neck T-Sh…
4	Microsoft Surface Pro 7
5	Poplin Shirt
6	Sealy Posturepedic Aspire

Total Quantity for the products shown

	pName	TotalQuantity
1	Sealy Posturepedic Aspire	700
2	Microsoft Surface Pro 7	600
3	Poplin Shirt	500
4	JBL speaker	500
5	Huawei P40	302
6	Men AIRism Crew Neck T-Sh…	300
7	iPhone X	110
8	Galaxy S9	110
9	Glass Cup	5

Total quantity for products with top 5 quantity

	pName	TotalQuantity
1	Sealy Posturepedic Aspire	700
2	Microsoft Surface Pro 7	600
3	Poplin Shirt	500
4	JBL speaker	500
5	Huawei P40	302
6	Men AIRism Crew Neck T-Sh…	300

Number of Unique purchases for Products vs Number of Unique user

	pName	NumUniquePurchases	NumUniqueUsers
1	Huawei P40	4	205
2	JBL speaker	5	205
3	Men AIRism Crew Neck T-Sh…	5	205
4	Microsoft Surface Pro 7	3	205
5	Poplin Shirt	5	205
6	Sealy Posturepedic Aspire	5	205

Query 9

Find products that are increasingly being purchased over at least 3 months.

Query 9:

Cross-verifying the output

```
---- Query 9 ----
-- Question: Find products that are increasingly being purchased over at least 3 months.

GO
-- create a view of the products sold, their quantities, month and year
CREATE VIEW ProductsInMonthYear AS
( SELECT pName, orderquantity, MONTH(orderdatetime) AS Month, YEAR(orderdatetime) AS Year
FROM ProductsInOrders JOIN Orders ON ProductsInOrders.OID=Orders.OID) ;

GO
-- view showing the total quantity of each product for per month, per year
CREATE VIEW PdtMonthlySales AS
(SELECT pName, Month, Year, SUM(orderquantity) AS TotalQuantity
FROM ProductsInMonthYear
GROUP BY pName, Month, Year);

GO
SELECT DISTINCT P1.pName
FROM PdtMonthlySales P1, PdtMonthlySales P2, PdtMonthlySales P3
WHERE(P1.pName=P2.pName AND P2.pName=P3.pName)
AND   ((P1.Year=P2.Year AND (P3.Year-P2.Year)=1 AND P1.Month=11 AND P2.Month=12 AND P3.Month=1) --Eg Nov 2019, Dec 2019 and Jan 2020
    OR((P2.Year-P1.Year)=1 AND P2.Year=P3.Year AND P1.Month=12 AND P2.Month=1 AND P3.Month=2) --Eg Dec 2019, Jan 2020 and Feb 2020
    OR(P1.Year=P2.Year AND P2.Year=P3.Year AND (P3.Month-P2.Month)=1 AND (P2.Month-P1.Month)=1)) --any 3 consecutive months in 2020.
AND (P3.TotalQuantity>P2.TotalQuantity AND P2.TotalQuantity>P1.TotalQuantity);

SELECT *
FROM PdtMonthlySales
ORDER BY pName, Month, Year;
```

	Pname	Month	Year	TotalQuantity
1	Adidas Cap	1	2020	3
2	Adidas Cap	2	2020	5
3	Adidas Cap	3	2020	5
4	Adidas Cap	4	2020	1
5	Adidas Cap	5	2020	5
6	Adidas Cap	6	2020	5
7	Adidas Cap	7	2020	4
8	Adidas Hoodie	1	2020	2
9	Adidas Hoodie	2	2020	4
10	Adidas Hoodie	3	2020	5
11	Adidas Hoodie	4	2020	2
12	Adidas Hoodie	5	2020	2
13	Adidas Hoodie	6	2020	7
14	Adidas Hoodie	7	2020	2
15	Dell XPS Notebook	3	2020	1
16	Dell XPS Notebook	6	2020	1
17	Galaxy S10	2	2020	1
18	Galaxy S10	3	2020	2
19	Galaxy S10	4	2020	1
20	Galaxy S10	5	2020	1

Output:

	pName
1	Adidas Hoodie
2	Galaxy S9
3	Huawei P40
4	iPhone X
5	iPhone XR
6	Men AIRism Crew Neck T-Sh…
7	Microsoft Surface Pro 7
8	Ultraboost Shoes

Thanks!

Any questions?

```sql
---- Query 2 ---- version 2
-- Question: Find products that received at least 100 ratings of "5" in August 2020, and order them by their average ratings.

-- Only extract those rows whereby the month of the feedback is August(08) and the year is 2020
WITH F0 AS ( --products that receive feedback in August 2020
SELECT *
FROM Feedback f
WHERE MONTH(feedbackDate) = 8 AND YEAR(feedbackDate) = 2020),

-- products that received ratings 5
F1 AS(
SELECT *
FROM F0
WHERE rating=5),

-- products with at least 100 ratings of 5
F2 AS(
SELECT pName, COUNT(rating) AS NumRatings5
FROM F1
GROUP BY pName
HAVING COUNT(rating)>=100)

-- First cast Rating to be a Float so we can get the decimal point values, then Average all the Rating values
-- Round the average rating to be 2 decimal point
SELECT F2.pName, ROUND(AVG(Cast(F0.rating as Float)),2) AS AvgRatings
FROM F2
JOIN F0 ON F2.pName=F0.pName
GROUP BY F2.pName
ORDER BY AvgRatings DESC;
```

	pName	AvgRatings
1	Galaxy S9	4.95
2	iPhone X	4.84

```
---- Query 3 ----
-- Question: For all products purchased in June 2020 that have been delivered, find the average time from the ordering date to the delivery date.

-- print average time of delivery in hours
SELECT CAST(AVG(DATEDIFF(second,orderdatetime, deliverydate)) AS FLOAT)/3600 AS AvgTimeOnDelivery
FROM Orders, ProductsInOrders
WHERE Orders.oID= ProductsInOrders.oID
AND orderdatetime >= '2020-06-01 00:00:01' AND orderdatetime <= '2020-06-30 23:59:59'
AND (deliverystatus = 'Delivered' OR deliverystatus = 'Returned');
```

	AvgTimeOnDelivery
1	116.3475

```
---- Query 4 ----
-- Question: Let us define the "latency" of an employee by the average that he/she takes to process a complaint.
--Find the employee with the smallest latency.

SELECT eID, AVG(DATEDIFF(second, filedatetime, handledatetime)) AS AvgLatency INTO LatencyRecord
FROM Complaint
GROUP BY eID;

-- cross-check table
SELECT *
FROM LatencyRecord;

SELECT eID
FROM LatencyRecord
WHERE AvgLatency = (SELECT MIN(AvgLatency) FROM LatencyRecord);
```

	eID	AvgLatency
1	1	2704800
2	2	2880000
3	3	2678400
4	4	2160000

	eID
1	4

```
---- Query 5 ----
-- Question: Produce a list that contains (i) all products made by Samsung, and
--(ii) for each of them, the number of shops on Sharkee that sell the product.

-- Part(i) --
SELECT pName
FROM Product
WHERE maker = 'Samsung';

-- Part(ii) -- version 1
SELECT Product.pName, COUNT(Sname) AS noOfShops
FROM ProductsInShops RIGHT JOIN Product ON Product.pName = ProductsInShops.pName
WHERE maker = 'Samsung'
GROUP BY Product.pName;

-- Part(ii) -- version 2
SELECT Product.pName AS Product, COUNT(ProductsInShops.PName) AS noOfShops
FROM Product
LEFT JOIN ProductsInShops
ON Product.pName = ProductsInShops.pName
WHERE maker = 'Samsung'
GROUP BY Product.pName;
```

	pName
1	Galaxy Note 10
2	Galaxy S10
3	Galaxy S10+
4	Galaxy S20
5	Galaxy S20+
6	Galaxy S9
7	Galaxy S9+
8	Samsung Galaxy Tab S5
9	Samsung LCD TV

	Product	noOfShops
1	Galaxy Note 10	0
2	Galaxy S10	1
3	Galaxy S10+	0
4	Galaxy S20	1
5	Galaxy S20+	1
6	Galaxy S9	1
7	Galaxy S9+	1
8	Samsung Galaxy Tab S5	1
9	Samsung LCD TV	1

```
---- Query 6 ----
-- Quesiton: Find shops that made the most revenue in August 2020.

WITH A1 AS
(
 SELECT t2.sName, SUM(t2.orderprice*t2.orderquantity) AS revenue
 FROM Orders as t1

 -- Left join on common attribute OrderID of both tables
 LEFT JOIN ProductsInOrders AS t2
 ON t1.oID = t2.oID

 -- OrderDateTime should fall under 2020/08
 WHERE MONTH(t1.orderdatetime) = 8 AND YEAR(t1.orderdatetime) = 2020
 -- Group by Shop name with aggregate function SUM of all revenue(OrderPrice*OrderQuantity) by this shop
 GROUP BY sName
)


SELECT sName
FROM A1
WHERE revenue = (SELECT MAX(revenue) FROM A1);
```

	sName
1	Walmart

	sName	revenue
1	iStudio	154000
2	Samsung	110000
3	Tesco	462799
4	Walmart	590525

```
---- Query 7 ----
-- Question: For users that made the most amount of complaints, find the most expensive products he/she has ever purchased.

-- Counts the total number of complaints each user has made
WITH A1 AS
(
 SELECT UserID, COUNT(UserID) as noOfComplaints
 FROM Complaint
 GROUP BY UserID
),

-- Select the users in A1 that has made the most complaints and their orderID
A2 AS
(
 SELECT t1.UserID, t2.oID
 FROM A1 as t1
 LEFT JOIN Orders as t2
 ON t1.UserID = t2.UserID
 WHERE noOfComplaints = (SELECT MAX(noOfComplaints) FROM A1)
),

-- Find all products that these users in A2 has ever purchased
A3 AS
(
 SELECT t1.UserID, t2.oID, t2.pName, t2.orderprice
 FROM A2 as t1
 LEFT JOIN ProductsInOrders as t2
 ON t1.oID = t2.oID

),

-- Find the most expensive product that each user in A3 has purchased
A4 AS
(
 SELECT UserID, MAX(OrderPrice) as maxProductPrice FROM A3
 GROUP BY UserID
)

-- Get the product name by matching UserID and the Product price
SELECT t1.UserID, t2.pName, t2.orderprice
FROM A4 as t1
LEFT JOIN A3 as t2
ON t1.UserID = t2.UserID AND t1.maxProductPrice = t2.OrderPrice;
```

	UserID	pName	orderprice
1	4	iPhone X	1400
2	87	Pureboost Shoes	179

```
---- Query 8 ----
-- Question: Find products that have never been purchased by some users, but are the top 5 most purchased products by other users in August 2020.

-- Create a view with all products sold in August, group by PName, and find total quantity
GO
CREATE VIEW AllProducts AS
(SELECT pName, SUM(orderquantity) AS TotalQuantity
FROM ProductsInOrders PIO
JOIN Orders O ON PIO.oID =O.oID AND
(O.OrderDateTime >= '2020.08.01 00:00:00' AND O.OrderDateTime < '2020.09.01 00:00:00')
GROUP BY PName);

GO
-- View that excludes top product in August
CREATE VIEW NonTop1Products AS
(SELECT *
FROM AllProducts AP
WHERE AP.TotalQuantity <> (SELECT MAX(TotalQuantity)
                           FROM Allproducts AP2));

GO
-- View that excludes top 2 product in August
CREATE VIEW NonTop2Products AS
(SELECT *
FROM NonTop1Products NT
WHERE NT.TotalQuantity <> (SELECT MAX(TotalQuantity)
                           FROM NonTop1Products NT1));

GO
-- View that excludes top 3 product in August
CREATE VIEW NonTop3Products AS
(SELECT *
FROM NonTop2Products NT
WHERE NT.TotalQuantity <> (SELECT MAX(TotalQuantity)
                           FROM NonTop2Products NT1));

GO
-- View that excludes top 4 product in August
CREATE VIEW NonTop4Products AS
(SELECT *
FROM NonTop3Products NT
WHERE NT.TotalQuantity <> (SELECT MAX(TotalQuantity)
                           FROM NonTop3Products NT1));
```

```
GO
-- View that excludes top 5 product in August
CREATE VIEW NonTop5Products AS
(SELECT *
FROM NonTop4Products NT
WHERE NT.TotalQuantity <> (SELECT MAX(TotalQuantity)
                           FROM NonTop4Products NT1));

GO
-- View that get top 5 product in August
-- Assume that there can be more than 5 products if products have the same order quantity.
CREATE VIEW TopProducts AS
(SELECT *
FROM AllProducts
EXCEPT
SELECT *
FROM NonTop5Products);

GO
-- View that gets the number of unique users
CREATE VIEW UserCount AS
SELECT COUNT(*) AS NumUniqueUsers
FROM Users;

GO
-- View that gets the number of unique purchases for each product
CREATE VIEW UniquePurchases AS
SELECT pName, Count(UserID) AS NumUniquePurchases
FROM (SELECT DISTINCT U.UserID, pName
      FROM Users U, Orders O ,ProductsInOrders PIO
      WHERE U.UserID=O.UserID AND O.oID=PIO.oID) AS UniquePurchase
GROUP BY pName;

GO
-- View that gets the products that are not bought by some users, but are top 5 products
SELECT DISTINCT TP.pName
FROM TopProducts TP, UserCount UC,UniquePurchases UP
WHERE TP.pName=UP.pName AND NumUniquePurchases < NumUniqueUsers;
```

```
GO

-- additional commands to visualise views
-- All products
SELECT *
FROM AllProducts
ORDER BY TotalQuantity DESC;

-- Top 5 products
SELECT *
FROM TopProducts
ORDER BY TotalQuantity DESC;

-- Number of unique users
SELECT *
FROM UserCount;

-- Number of  unique purchases for each product
SELECT *
FROM UniquePurchases;

-- Number of  unique purchases for each product with number of unique users added
SELECT TP.pName, NumUniquePurchases, NumUniqueUsers
FROM TopProducts TP, UserCount UC,UniquePurchases UP
WHERE TP.pName=UP.pName;
```

	pName
1	Huawei P40
2	JBL speaker
3	Men AIRism Crew Neck T-Sh…
4	Microsoft Surface Pro 7
5	Poplin Shirt
6	Sealy Posturepedic Aspire

	pName	TotalQuantity
1	Sealy Posturepedic Aspire	700
2	Microsoft Surface Pro 7	600
3	Poplin Shirt	500
4	JBL speaker	500
5	Huawei P40	302
6	Men AIRism Crew Neck T-Sh…	300
7	iPhone X	110
8	Galaxy S9	110
9	Glass Cup	5

	pName	TotalQuantity
1	Sealy Posturepedic Aspire	700
2	Microsoft Surface Pro 7	600
3	Poplin Shirt	500
4	JBL speaker	500
5	Huawei P40	302
6	Men AIRism Crew Neck T-Sh…	300

	pName	NumUniquePurchases	NumUniqueUsers
1	Huawei P40	4	205
2	JBL speaker	5	205
3	Men AIRism Crew Neck T-Sh…	5	205
4	Microsoft Surface Pro 7	3	205
5	Poplin Shirt	5	205
6	Sealy Posturepedic Aspire	5	205

```
---- Query 9 ----
-- Question: Find products that are increasingly being purchased over at least 3 months.

GO
-- create a view of the products sold, their quantities, month and year
CREATE VIEW ProductsInMonthYear AS
( SELECT pName, orderquantity, MONTH(orderdatetime) AS Month, YEAR(orderdatetime) AS Year
FROM ProductsInOrders JOIN Orders ON ProductsInOrders.OID=Orders.OID) ;

GO
-- view showing the total quantity of each product for per month, per year
CREATE VIEW PdtMonthlySales AS
(SELECT pName, Month, Year, SUM(orderquantity) AS TotalQuantity
FROM ProductsInMonthYear
GROUP BY pName, Month, Year);

GO
SELECT DISTINCT P1.pName
FROM PdtMonthlySales P1, PdtMonthlySales P2, PdtMonthlySales P3
WHERE(P1.pName=P2.pName AND P2.pName=P3.pName)
AND   ((P1.Year=P2.Year AND (P3.Year-P2.Year)=1 AND P1.Month=11 AND P2.Month=12 AND P3.Month=1) --Eg Nov 2019, Dec 2019 and Jan 2020
    OR((P2.Year-P1.Year)=1 AND P2.Year=P3.Year AND P1.Month=12 AND P2.Month=1 AND P3.Month=2) --Eg Dec 2019, Jan 2020 and Feb 2020
    OR(P1.Year=P2.Year AND P2.Year=P3.Year AND (P3.Month-P2.Month)=1 AND (P2.Month-P1.Month)=1)) --any 3 consecutive months in 2020.
AND (P3.TotalQuantity>P2.TotalQuantity AND P2.TotalQuantity>P1.TotalQuantity);

SELECT *
FROM PdtMonthlySales
ORDER BY pName, Month, Year;
```

	pName
1	Adidas Hoodie
2	Galaxy S9
3	Huawei P40
4	iPhone X
5	iPhone XR
6	Men AIRism Crew Neck T-Sh…
7	Microsoft Surface Pro 7
8	Ultraboost Shoes

	Pname	Month	Year	TotalQuantity
1	Adidas Cap	1	2020	3
2	Adidas Cap	2	2020	5
3	Adidas Cap	3	2020	5
4	Adidas Cap	4	2020	1
5	Adidas Cap	5	2020	5
6	Adidas Cap	6	2020	5
7	Adidas Cap	7	2020	4
8	Adidas Hoodie	1	2020	2
9	Adidas Hoodie	2	2020	4
10	Adidas Hoodie	3	2020	5
11	Adidas Hoodie	4	2020	2
12	Adidas Hoodie	5	2020	2
13	Adidas Hoodie	6	2020	7
14	Adidas Hoodie	7	2020	2
15	Dell XPS Notebook	3	2020	1
16	Dell XPS Notebook	6	2020	1
17	Galaxy S10	2	2020	1
18	Galaxy S10	3	2020	2
19	Galaxy S10	4	2020	1
20	Galaxy S10	5	2020	1

Triggers

- Here you have a list of items
- And some text
- But remember not to overload your slides with content

Your audience will listen to you or read the content, but won’t do both.

Big concept

Bring the attention of your audience over a key concept using icons or illustrations

You can also split your content

White

Is the color of milk and fresh snow, the color produced by the combination of all the colors of the visible spectrum.

Black

Is the color of ebony and of outer space. It has been the symbolic color of elegance, solemnity and authority.

In two or three columns

Yellow

Is the color of gold, butter and ripe lemons. In the spectrum of visible light, yellow is found between green and orange.

Blue

Is the colour of the clear sky and the deep sea. It is located between violet and green on the optical spectrum.

Red

Is the color of blood, and because of this it has historically been associated with sacrifice, danger and courage.

EDIT IN GOOGLE SLIDES

Click on the button under the presentation preview that says "Use as Google Slides Theme".

You will get a copy of this document on your Google Drive and will be able to edit, add or delete slides.

You have to be signed in to your Google account.

EDIT IN POWERPOINT®

Click on the button under the presentation preview that says "Download as PowerPoint template". You will get a .pptx file that you can edit in PowerPoint.

Remember to download and install the fonts used in this presentation (you'll find the links to the font files needed in the Presentation design slide)

More info on how to use this template at www.slidescarnival.com/help-use-presentation-template

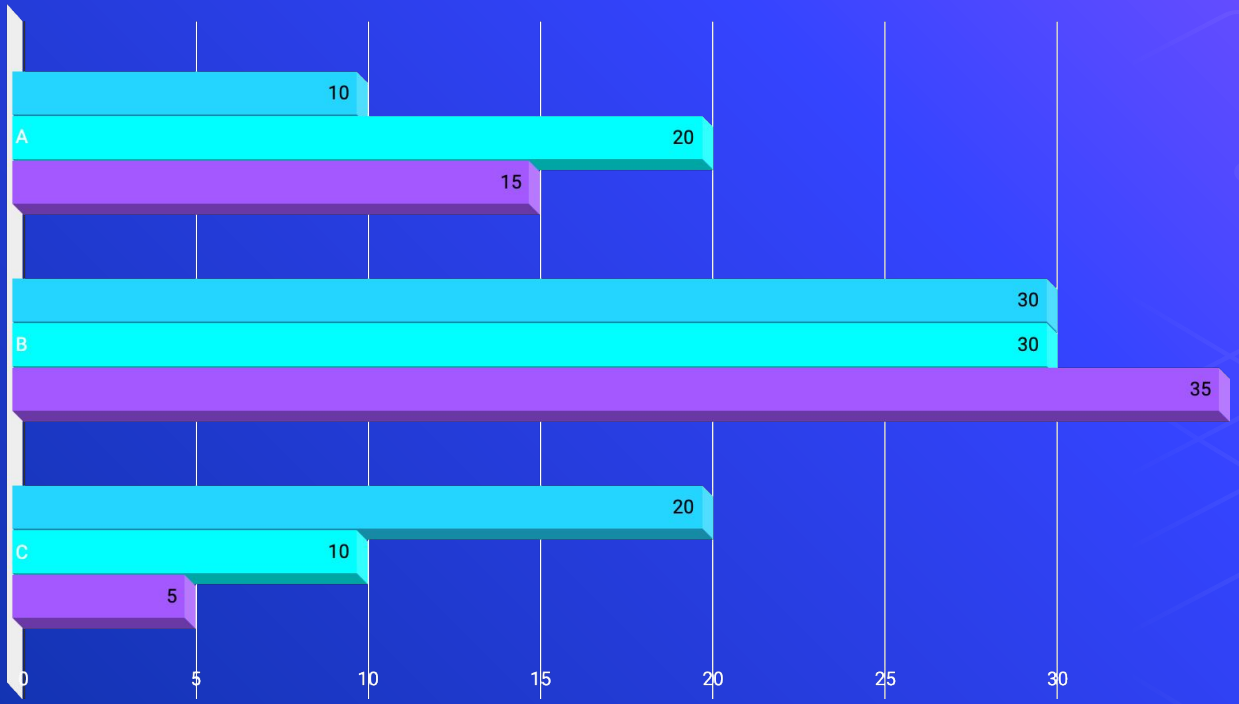
This template is free to use under Creative Commons Attribution license. You can keep the Credits slide or mention SlidesCarnival and other resources used in a slide footer.

A picture is worth a thousand words

A complex idea can be conveyed with just a single still image, namely making it possible to absorb large amounts of data quickly.

> Quotations are commonly printed as a means of inspiration and to invoke philosophical thoughts from the reader.


Want big impact?
Use big image.



You can insert graphs from Google Sheets

Use diagrams to explain your ideas

And tables to compare data

	A	B	C
Yellow	10	20	7
Blue	30	15	10
Orange	5	24	16

Hello!

I am Jayden Smith

I am here because I love to give presentations.

You can find me at @username

Maps

Find more maps at slidescarnival.com/extra-free-resources-icons-and-maps

89,526,124

Whoa! That’s a big number, aren’t you proud?

89,526,124$

That’s a lot of money

185,244 users

And a lot of users

100%

Total success!

Our process is easy

1

Vestibulum congue tempus

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor. Donec facilisis lacus eget mauris.

2

Vestibulum congue tempus

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor. Donec facilisis lacus eget mauris.

3

Vestibulum congue tempus

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor. Donec facilisis lacus eget mauris.

Let’s review some concepts

Yellow

Is the color of gold, butter and ripe lemons. In the spectrum of visible light, yellow is found between green and orange.

Blue

Is the colour of the clear sky and the deep sea. It is located between violet and green on the optical spectrum.

Red

Is the color of blood, and because of this it has historically been associated with sacrifice, danger and courage.

Yellow

Is the color of gold, butter and ripe lemons. In the spectrum of visible light, yellow is found between green and orange.

Blue

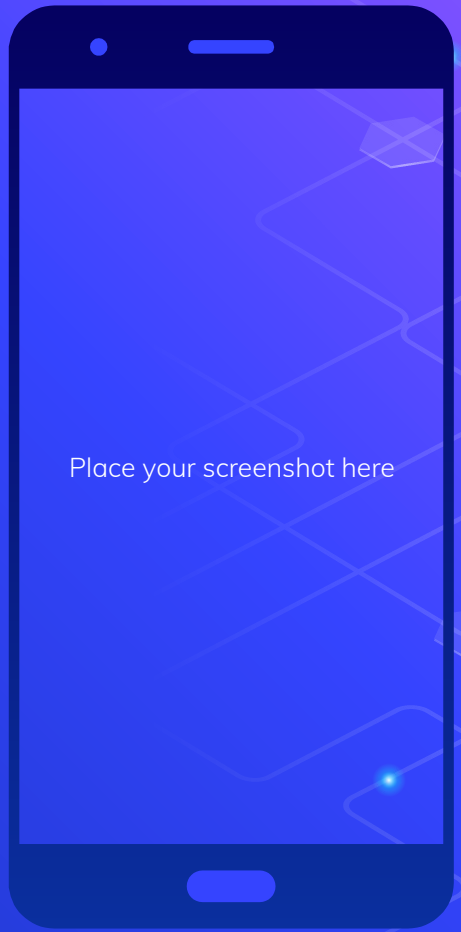
Is the colour of the clear sky and the deep sea. It is located between violet and green on the optical spectrum.

Red

Is the color of blood, and because of this it has historically been associated with sacrifice, danger and courage.

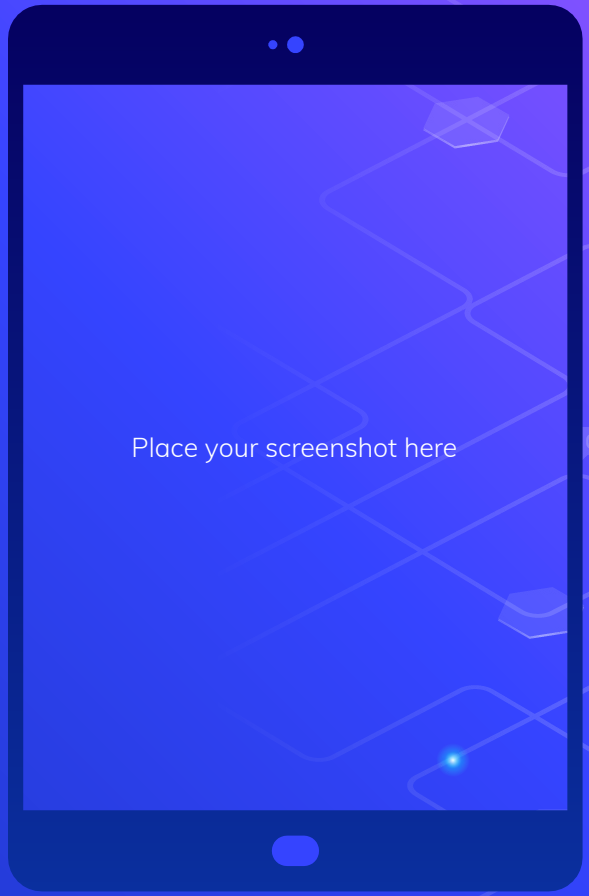
Mobile project

Show and explain your web, app or software projects using these gadget templates.



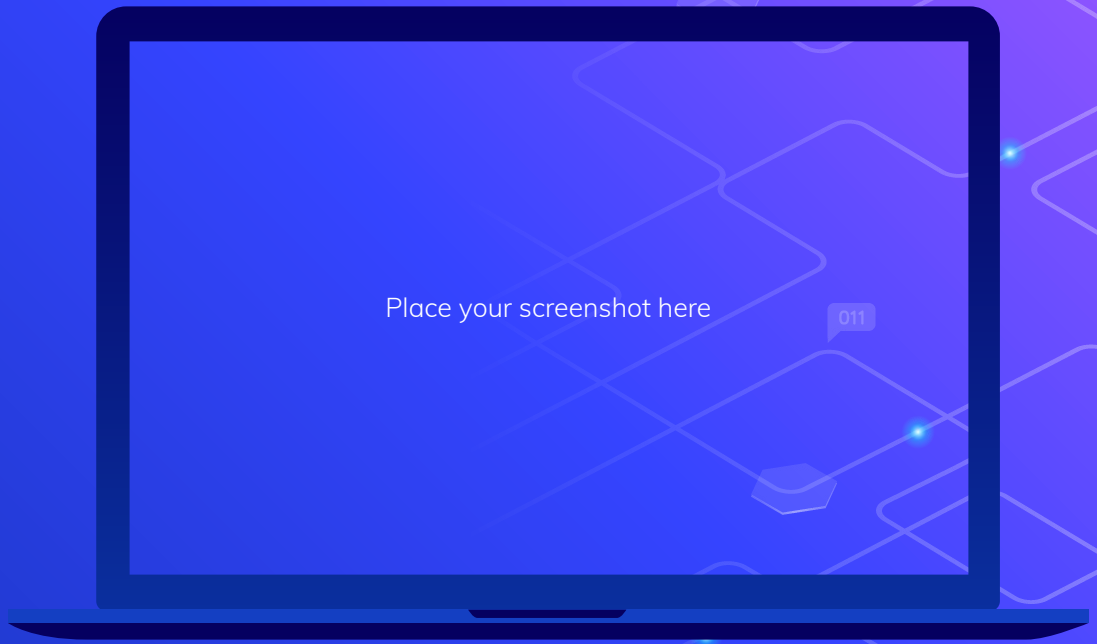
Tablet project

Show and explain your web, app or software projects using these gadget templates.



Desktop project

Show and explain your web, app or software projects using these gadget templates.



Thanks!

Any questions?

You can find me at:

@username

user@mail.me

Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash

Presentation design

This presentation uses the following typographies:

- Titles: Lexend Deca
- Body copy: Muli light

Download for free at:

https://www.lexend.com/

https://www.fontsquirrel.com/fonts/muli

You don't need to keep this slide in your presentation. It's only here to serve you as a design guide if you need to create new slides or download the fonts to edit the presentation in PowerPoint®

Extra resources

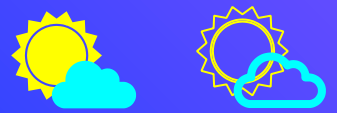
SlidesCarnival icons are editable shapes.

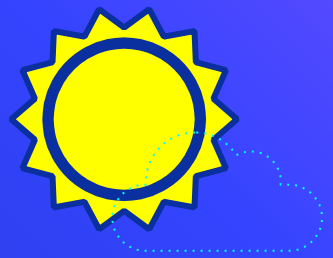
This means that you can:

- Resize them without losing quality.
- Change fill color and opacity.
- Change line color, width and style.

Isn't that nice? :)

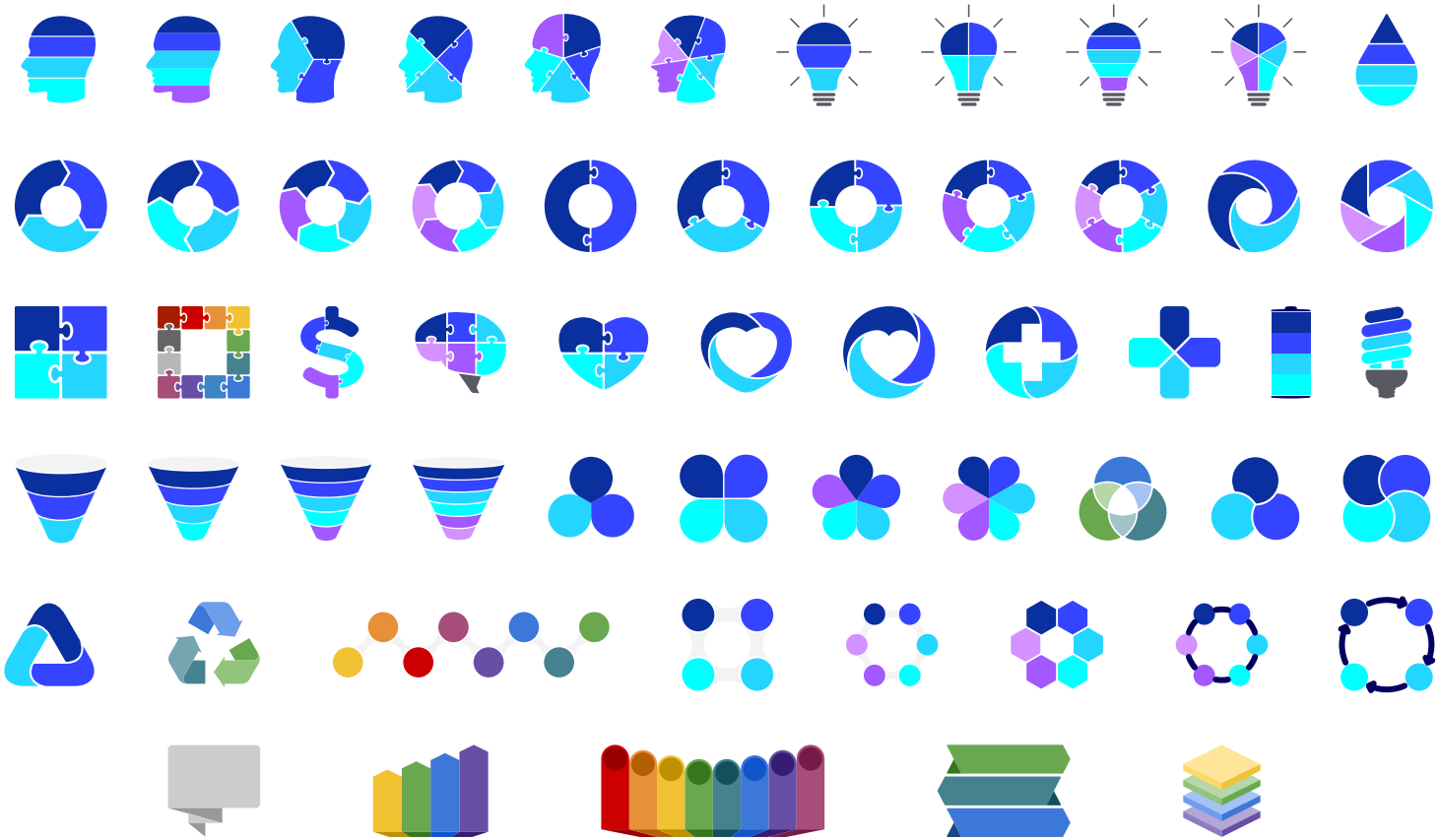
Examples:





Find more icons at
slidescarnival.com/extra-free-resources-icons-and-maps

Diagrams and infographics



You can also use any emoji as an icon!
And of course it resizes without losing quality.

How? Follow Google instructions
https://twitter.com/googledocs/status/730087240156643328

and many more...

Free templates for all your presentation needs

For PowerPoint and Google Slides

100% free for personal or commercial use

Ready to use, professional and customizable

Blow your audience away with attractive visuals