-- 1) Most generous organizations on average

```
WITH AvgOrganizationDonation AS (
   SELECT OrganizationID, ROUND(AVG(Amount), 2) as avg_donation_amt
   FROM Donations
   GROUP BY OrganizationID
)

SELECT d.OrganizationID, ad.avg_donation_amt
FROM Donations d
JOIN AvgOrganizationDonation ad
ON d.OrganizationID = ad.OrganizationID
GROUP BY d.OrganizationID, ad.avg_donation_amt
HAVING ad.avg_donation_amt > (SELECT AVG(Amount) FROM Donations WHERE OrganizationID
IS NULL);
```

-- 2) Individuals most willing to donate compared to organizations

```
WITH TotalPersonDonation AS (
   SELECT PersonID, ROUND(SUM(Amount), 2) as total_person_donation
   FROM Donations
   GROUP BY PersonID
),
TotalIndividualOrgDonation AS (
   SELECT OrganizationID, ROUND(SUM(Amount), 2) as total_org_donation
   FROM Donations
   GROUP BY OrganizationID
)

SELECT d.PersonID, tpd.total_person_donation
FROM Donations d
JOIN TotalPersonDonation tpd
ON d.PersonID = tpd.PersonID
JOIN TotalIndividualOrgDonation tod
ON d.OrganizationID = tod.OrganizationID
GROUP BY d.PersonID, tpd.total_person_donation, tod.total_org_donation
HAVING tpd.total_person_donation > tod.total_org_donation;
```

-- 3) Events with the highest total donations

```
SELECT TOP 10 SUM(d.Amount) as total_donation, e.EventID, e.Description
FROM Donations d
JOIN Event e
ON d.EventID = e.EventID
```

```
GROUP BY e.EventID, e.Description
ORDER BY total_donation DESC;
```

-- 4) Events where people donated more than organizations

```
WITH EventPersonDonation AS (
   SELECT EventID, ROUND(SUM(Amount), 2) as person_donation
   FROM Donations
   WHERE PersonID IS NOT NULL
   GROUP BY PersonID, EventID
),
EventOrgDonation AS (
   SELECT EventID, ROUND(SUM(Amount), 2) as org_donation
   FROM Donations
   WHERE OrganizationID IS NOT NULL
   GROUP BY OrganizationID, EventID
)

SELECT d.EventID, e.Description
FROM Donations d
LEFT JOIN EventPersonDonation epd
ON d.EventID = epd.EventID
LEFT JOIN EventOrgDonation eod
ON d.EventID = eod.EventID
LEFT JOIN Event e
ON e.EventID = d.EventID
GROUP BY d.PersonID, epd.person_donation, eod.org_donation, e.Description
HAVING epd.person_donation > eod.org_donation;
```

-- 5) Events that were more expensive than the amount they received in donations

```
WITH EventOverallCost AS (
  SELECT EventID, (Food_Cost + Venue_Cost + Speaker_Cost) AS event_total_cost
  FROM Costs
),
EventTotalDonation AS (
  SELECT EventID, SUM(Amount) as event_donation_amt
  FROM Donations
  GROUP BY EventID
)

SELECT c.EventID, eoc.event_total_cost, etd.event_donation_amt
```

```
FROM Costs c
JOIN EventOverallCost eoc
ON c.EventID = eoc.EventID
JOIN EventTotalDonation etd
ON c.EventID = etd.EventID
GROUP BY c.EventID, eoc.event_total_cost, etd.event_donation_amt
HAVING eoc.event_total_cost > etd.event_donation_amt;
```

-- 6) The most cost-effective events that were conferences, epos, concerts, and or fairs

```
WITH EventOverallCost AS (
  SELECT EventID, (Food_Cost + Venue_Cost + Speaker_Cost) AS event_total_cost
  FROM Costs
),
EventTotalDonation AS (
  SELECT e.EventID, SUM(Amount) as event_donation_amt, e.Description
  FROM Donations d
  LEFT JOIN Event e
  ON d.EventId = e.EventID
  WHERE e.Description LIKE '%Conference%'OR e.Description LIKE '%Expo%'OR
e.Description LIKE '%Concert%'OR e.Description LIKE'%Fair%'
  GROUP BY e.EventID, e.Description
)

SELECT c.EventID, eoc.event_total_cost, etd.event_donation_amt, etd.Description
FROM Costs c
JOIN EventOverallCost eoc
ON c.EventID = eoc.EventID
JOIN EventTotalDonation etd
ON c.EventID = etd.EventID
GROUP BY c.EventID, eoc.event_total_cost, etd.event_donation_amt, etd.Description
HAVING eoc.event_total_cost < etd.event_donation_amt;
```

-- 7) The speakers who are the most recurring that charity events

```
SELECT p.FirstName, p.LastName, s.PersonID, COUNT(s.PersonID) AS num_spoken
FROM Speaker s
JOIN Person p
ON s.PersonID = p.PersonID
GROUP BY p.FirstName, p.LastName, s.PersonID;
```

```sql
-- 8) The events that had more people willing to speak for the event
SELECT s.EventID, COUNT(s.EventID) AS num_speakers, e.Description
FROM Speaker s
JOIN Event e
ON s.EventID = e.EventID
GROUP BY s.EventID, e.Description
HAVING COUNT(s.EventID) > 1;


-- 9) The people that are most willing to participate.
SELECT p.FirstName, p.LastName, ea.PersonID, COUNT(ea.PersonID) AS num_attended
FROM Event_Attendance ea
JOIN Person p
ON ea.PersonID = p.PersonID
GROUP BY ea.PersonID, p.FirstName, p.LastName
HAVING COUNT(ea.PersonID) > 2
ORDER BY num_attended;


-- 10) The organizations that are most philanthropic
SELECT o.Name, p.OrganizationID, o.Description, COUNT(ea1.PersonID) AS num_attended
FROM Event_Attendance ea1
JOIN Person p
ON ea1.PersonID = p.PersonID
JOIN Organization o
ON o.OrganizationID = p.OrganizationID
GROUP BY p.FirstName, p.LastName, p.OrganizationID, o.Name, o.Description
HAVING COUNT(ea1.PersonID) = (
   SELECT MAX(attend_num)
   FROM (
       SELECT PersonID, COUNT(PersonID) AS attend_num
       FROM Event_Attendance
       GROUP BY PersonID
   ) AS inner_counts
)
ORDER BY num_attended;

-- 11) Organizations most involved when it comes to speaking
SELECT o.Name, p.OrganizationID, o.Description, s.PersonID, COUNT(s.PersonID) AS
num_attended
FROM Speaker s
```

```
JOIN Person p
ON s.PersonID = p.PersonID
JOIN Organization o
ON o.OrganizationID = p.OrganizationID
GROUP BY s.PersonID, p.FirstName, p.LastName, p.OrganizationID, o.Name, o.Description
HAVING COUNT(s.PersonID) = (
   SELECT MAX(attend_num)
   FROM (
       SELECT PersonID, COUNT(PersonID) AS attend_num
       FROM Speaker
       GROUP BY PersonID
   ) AS inner_counts
)
ORDER BY num_attended;
```

-- 12) Organizations least involved when it comes to speaking

```
SELECT o.Name, p.OrganizationID, o.Description, s.PersonID, COUNT(s.PersonID) AS
num_attended
FROM Speaker s
JOIN Person p
ON s.PersonID = p.PersonID
JOIN Organization o
ON o.OrganizationID = p.OrganizationID
GROUP BY s.PersonID, p.FirstName, p.LastName, p.OrganizationID, o.Name, o.Description
HAVING COUNT(s.PersonID) = (
   SELECT MIN(attend_num)
   FROM (
       SELECT PersonID, COUNT(PersonID) AS attend_num
       FROM Speaker
       GROUP BY PersonID
   ) AS inner_counts
)
ORDER BY num_attended;
```

-- 13) Total Cost CTE

```
WITH Total_Costs AS (
SELECT CostID, EventID, Food_Cost, Venue_Cost, Speaker_Cost,
      Food_Cost + Venue_Cost + Speaker_Cost AS Total_Cost
FROM Costs)

SELECT e.EventID, e.Description, tc.Total_Cost
FROM Event e JOIN Total_Costs tc
ON e.EventID = tc.EventID
WHERE Total_Cost > 1000;
```

```sql
-- 14) Popular Venues
SELECT v.VenueID, v.Name, COUNT(e.EventID) AS Num_Events
FROM Venue v JOIN Event e
ON v.VenueID = e.VenueID
GROUP BY v.VenueID, v.Name
HAVING COUNT(e.EventID) > 1
ORDER BY Num_Events DESC;

-- 15) Best Venue Location
SELECT City, COUNT(VenueID) AS Num_Venues
FROM Venue
GROUP BY City
HAVING COUNT(VenueID) > 1
ORDER BY Num_Venues DESC;

-- 16) High-charging Venues
SELECT TOP 10 v.Name, c.Venue_Cost
FROM Venue v JOIN Event e
ON v.VenueID = e.VenueID
JOIN Costs c
ON e.EventID = c.EventID
ORDER BY c.Venue_Cost DESC;

-- 17) Average Venue Costs
SELECT v.VenueID, v.Name, AVG(c.Venue_Cost) AS Avg_Cost
FROM Venue v JOIN Event e
ON v.VenueID = e.VenueID
JOIN Costs c
ON e.EventID = c.EventID
WHERE v.Name IN (SELECT v.Name
      FROM Venue v JOIN Event e
      ON v.VenueID = e.VenueID
      GROUP BY v.VenueID, v.Name
      HAVING COUNT(e.EventID) > 1)
GROUP BY v.VenueID, v.Name
HAVING AVG(c.Venue_Cost) > 500
ORDER BY Avg_Cost DESC;

-- 18) High-charging Speakers
SELECT TOP 10 p.PersonID, p.FirstName, p.LastName,
      e.EventID, c.Speaker_Cost
FROM Person p JOIN Speaker s
ON p.PersonID = s.PersonID
JOIN Event e
ON s.EventID = e.EventID
JOIN Costs c
ON e.EventID = c.EventID
ORDER BY c.Speaker_Cost DESC;

-- 19) Organization with Multiple Sponsors--
SELECT
```

```sql
    o.Name As [Organization Name],
    o.Email,
    o.CauseType,
    COUNT(s.SponsorsID) AS num_of_sponsors
FROM Organization o
JOIN Sponsors s
ON o.OrganizationID = s.OrganizationID
GROUP BY o.Name, o.Email, o.CauseType
HAVING COUNT(s.SponsorsID) > 1
ORDER BY num_of_sponsors DESC;

-- 20) TOP 10 Donation Organizations--
SELECT TOP 10
    o.Name As [Organization Name],
    o.Email,
    o.CauseType,
    SUM(d.Amount) AS total_donation
FROM Organization o
JOIN Donations d
ON o.OrganizationID = d.OrganizationID
GROUP BY o.Name, o.Email, o.CauseType
ORDER BY total_donation DESC;

-- 21) Organizations' Donation Probabilities --
SELECT
    o.Name As [Organization Name], o.Email, o.CauseType,
    SUM(d.Amount) AS Total_Organization_Donation,
    SUM(d.Amount)/(SELECT SUM(Amount) FROM Donations)*100
    AS [probability of donation]
FROM Organization o
JOIN Donations d
ON o.OrganizationID = d.OrganizationID
GROUP BY o.Name, o.Email, o.CauseType
ORDER BY [probability of donation] DESC;

-- 22) Event Cost Percentage --
WITH overall_cost AS (
    SELECT
        EventID,
        Food_Cost + Venue_Cost + Speaker_Cost AS total_cost
    FROM Costs
)
SELECT
```

```
    c.eventID, e.Description,
    c.Food_Cost/oc.total_cost *100 AS prob_food_cost,
    c.Venue_Cost/oc.total_cost *100 AS prob_venue_cost,
    c.Speaker_Cost/oc.total_cost *100 AS prob_speaker_cost,
    oc.total_cost
FROM Costs c
JOIN Event e ON c.eventID = e.eventID
JOIN overall_cost oc ON oc.EventID = c.EventID;

- 23) High Average Attendees Events --
WITH EventAttendanceCounts AS (
    SELECT EventID, COUNT(PersonID) AS AttendanceCount
    FROM Event_Attendance
    GROUP BY EventID
)
SELECT
    e.EventID,
    e.VenueID,
    e.Description,
    ROUND(AVG(eac.AttendanceCount), 0) AS avg_attendance
FROM Event e
JOIN EventAttendanceCounts eac ON e.EventID = eac.EventID
GROUP BY
    e.EventID,
    e.VenueID,
    e.Description
HAVING AVG(eac.AttendanceCount) >
    (SELECT AVG(CAST(AttendanceCount AS FLOAT))
     FROM EventAttendanceCounts)
ORDER BY avg_attendance DESC;

- 24) Seeking Donation Amount from desired Organization--
-- Define function
CREATE FUNCTION dbo.fnOrganizationDonation_VKN(@Organization_ID INT)
RETURNS INT
BEGIN
    DECLARE @TotalDonation INT;
    SELECT @TotalDonation = SUM(d.Amount)
    FROM Donations d
    JOIN Organization o ON d.OrganizationID = o.OrganizationID
    WHERE o.OrganizationID = @Organization_ID;
    RETURN ISNULL(@TotalDonation, 0);
END;
```

```
GO

-- Call function
SELECT
 o.OrganizationID, o.Name,
 dbo.fnOrganizationDonation_VKN(o.OrganizationID) AS sum_donation
FROM Donations d
JOIN Organization o ON d.OrganizationID = o.OrganizationID
WHERE o.OrganizationID = 613 OR o.OrganizationID = 623;
```