Stored procedure:

```
DELIMITER $$

CREATE PROCEDURE GetSuitableVendors (
    IN MaxBasePrice DECIMAL(10, 2),
    IN DesiredServiceCategory VARCHAR(50)
)
BEGIN
    IF MaxBasePrice <= 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Maximum base price must be greater than 0';
    END IF;

    SELECT VendorName, ServiceCategory, BasePrice, Description
    FROM Vendors
    WHERE BasePrice <= MaxBasePrice
      AND ServiceCategory = DesiredServiceCategory
    ORDER BY BasePrice ASC;
END $$

DELIMITER ;

CALL GetSuitableVendors(1000, "Food");
```

Constraint:

```
ALTER TABLE Events
ADD CONSTRAINT chk_budget_positive CHECK (Budget >= 0);
```

Constraint:

```
ALTER TABLE Venues
ADD CONSTRAINT max_capacity_check CHECK (MaxCapacity >= 10);
```

Constraint:

```
ALTER TABLE Events
ADD CONSTRAINT unique_venue_date UNIQUE (VenueID, EventDate);
```

Trigger:

```
DELIMITER $$

CREATE TRIGGER ValidateReviewInsertion
BEFORE INSERT ON Reviews
FOR EACH ROW
BEGIN

  IF NEW.Rating < 1 OR NEW.Rating > 5 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Rating must be between 1 and 5';
  END IF;

  IF NOT EXISTS (
    SELECT 1
    FROM Services
    WHERE ServiceID = NEW.ServiceID
  ) THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'The service ID does not exist';
  END IF;

  IF NEW.ReviewDate > CURDATE() THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Review date cannot be in the future';
  END IF;
END $$
```

```
DELIMITER ;
```

Transaction:

```
DELIMITER $$

CREATE PROCEDURE AddNewEvent (
    IN p_event_id INT,
    IN p_event_name VARCHAR(100),
    IN p_venue_id INT,
    IN p_organizer_id INT,
    IN p_event_type VARCHAR(50),
    IN p_event_date DATE,
    IN p_budget DECIMAL(10, 2),
    IN p_description VARCHAR(500)
)
BEGIN
    DECLARE is_venue_available INT;
    DECLARE is_vendor_available INT;

    START TRANSACTION;

    IF p_budget <= 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Budget must be greater than 0';
    END IF;

    SELECT COUNT(*) INTO is_venue_available
    FROM Events
    WHERE VenueID = p_venue_id AND EventDate = p_event_date;

    IF is_venue_available > 0 THEN
```

```
        ROLLBACK;
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'The venue is already booked for the selected date';
    END IF;

    SELECT COUNT(DISTINCT v.VendorID) INTO is_vendor_available
    FROM Vendors v
    LEFT JOIN Services s ON v.VendorID = s.VendorID
    LEFT JOIN Events e ON e.EventID = s.ServiceID
    WHERE e.EventDate != p_event_date OR e.EventDate IS NULL;

    IF is_vendor_available = 0 THEN
        ROLLBACK;
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No suitable vendors available for the event date';
    END IF;

    INSERT INTO Events (EventID, EventName, VenueID, OrganizerID, EventType, EventDate, Budget,
Description)
    VALUES (p_event_id, p_event_name, p_venue_id, p_organizer_id, p_event_type, p_event_date,
p_budget, p_description);

    COMMIT;
END $$

DELIMITER ;
```

Advanced query:

```
SELECT
    v1.VendorName AS Vendor1,
    v2.VendorName AS Vendor2,
    v1.ServiceCategory,
```

```
    AVG(r1.Rating) AS Vendor1AverageRating,
    AVG(r2.Rating) AS Vendor2AverageRating,
    ABS(AVG(r1.Rating) - AVG(r2.Rating)) AS RatingDifference
FROM
    Vendors v1
JOIN
    Services s1 ON v1.VendorID = s1.VendorID
JOIN
    Reviews r1 ON s1.ServiceID = r1.ServiceID
JOIN
    Vendors v2 ON v1.ServiceCategory = v2.ServiceCategory AND v1.VendorID != v2.VendorID
JOIN
    Services s2 ON v2.VendorID = s2.VendorID
JOIN
    Reviews r2 ON s2.ServiceID = r2.ServiceID
GROUP BY
    v1.VendorID, v2.VendorID
HAVING
    RatingDifference <= 0.5
ORDER BY
    v1.ServiceCategory, RatingDifference
LIMIT 15;
```

Advanced query:

```
SELECT
    v.VenueName,
    COUNT(e.EventID) AS TotalEvents,
    AVG(e.Budget) AS AverageBudget
FROM
    Venues v
JOIN
```

```
    Events e ON v.VenueID = e.VenueID
WHERE
    v.MaxCapacity > 100
    AND e.EventDate >= '2024-09-17'
GROUP BY
    v.VenueID
ORDER BY
    TotalEvents DESC
LIMIT 15;
```

Advanced Query:

```
SELECT
    v.VenueName,
    v.Address,
    AVG(r.Rating) AS AvgRating,
    COUNT(r.ReviewID) AS ReviewCount,
    SUM(s.Price) AS TotalServiceCost,
    (SELECT COUNT(*)
     FROM Events e
     JOIN Users u ON e.OrganizerID = u.UserID
     WHERE e.VenueID = v.VenueID
       AND e.EventDate >= '2024-01-01'
    ) AS RecentOrganizerEvents
FROM
    Venues v
JOIN
    Reviews r ON v.VenueID = r.VenueID
JOIN
    Services s ON s.BundleID IN (
        SELECT BundleID
        FROM ServiceBundles
```

```
        WHERE BundlePrice > 50
    )
WHERE
    v.MaxCapacity > 50
    AND r.ReviewDate >= '2024-09-10'
    AND r.Rating >= 2
GROUP BY
    v.VenueID
HAVING
    AVG(r.Rating) >= 2.0
    AND RecentOrganizerEvents >= 2
    AND TotalServiceCost > 100
ORDER BY
    AvgRating DESC,
    ReviewCount DESC
LIMIT 15;
```