

CIS 9340 Group 1 Data Application

1. Creating the Database with SQL

```
CREATE TABLE Artists(
ArtistID NUMBER NOT NULL,
Name VARCHAR(20),
Genre VARCHAR(20),
Country VARCHAR(20),
ContactInfo VARCHAR(20),
Agent VARCHAR(20),
CONSTRAINT pk_Artists PRIMARY KEY (ArtistID))
```

```
CREATE TABLE Merchandise(
MerchID NUMBER NOT NULL,
ItemDate DATE,
Price NUMBER NOT NULL,
StockAvailable VARCHAR(20),
CONSTRAINT pk_Merchandise PRIMARY KEY (MerchID))
```

```
CREATE TABLE MerchSales(
MerchSaleID NUMBER NOT NULL,
Quantity NUMBER NOT NULL,
TotalAmount NUMBER NOT NULL,
CONSTRAINT pk_MerchSales PRIMARY KEY (MerchSaleID)
)
```

```
CREATE TABLE Merchandise_MerchSales(
MerchID NUMBER NOT NULL,
MerchSaleID NUMBER NOT NULL,
CONSTRAINT pk_Merchandise_MerchSales PRIMARY KEY (MerchID, MerchSaleID)
)
```

```
CREATE TABLE Attendees(
AttendeeID NUMBER NOT NULL,
Name VARCHAR(20),
Email VARCHAR(20),
Age NUMBER NOT NULL,
CheckInStatus VARCHAR(20),
CONSTRAINT pk_Attendees PRIMARY KEY (AttendeeID)
)
```

```
CREATE TABLE MerchSales_Attendees(
AttendeeID NUMBER NOT NULL,
```

```
MerchSaleID NUMBER NOT NULL,  
CONSTRAINT pk_MerchSales_Attendees PRIMARY KEY (AttendeeID,MerchSaleID)  
)
```

```
CREATE TABLE Attendees_Tickets(  
AttendeeID NUMBER NOT NULL,  
Name VARCHAR(20),  
Email VARCHAR(20),  
Age NUMBER NOT NULL,  
CheckInStatus VARCHAR(20),  
TicketID NUMBER NOT NULL,  
Type VARCHAR(20),  
Price NUMBER NOT NULL,  
Availability VARCHAR(20),  
SaleDate DATE,  
CONSTRAINT pk_Attendees_Tickets PRIMARY KEY (AttendeeID)  
)
```

```
CREATE TABLE Tickets(  
TicketID NUMBER NOT NULL,  
Type VARCHAR(20),  
Price NUMBER NOT NULL,  
Availability VARCHAR(20),  
SaleDate DATE,  
VendorID NUMBER NOT NULL,  
CONSTRAINT pk_Tickets PRIMARY KEY (TicketID))
```

```
CREATE TABLE Vendor(  
VendorID NUMBER NOT NULL,  
Name VARCHAR(20),  
Category VARCHAR(20),  
ContactInfo VARCHAR(20),  
LocationAssigned VARCHAR(20),  
CONSTRAINT pk_Vendor PRIMARY KEY (VendorID)  
)
```

```
CREATE TABLE VendorSales(  
SaleID NUMBER NOT NULL,  
SaleDate DATETIME,  
ItemSold VARCHAR(20),  
Quantity NUMBER NOT NULL,  
Amount NUMBER NOT NULL,  
VendorID NUMBER NOT NULL,
```

```
CONSTRAINT pk_VendorSales PRIMARY KEY (SaleID))
```

```
CREATE TABLE Vendors_MerchSales(
    VendorID NUMBER NOT NULL,
    MerchSaleID NUMBER NOT NULL,
    CONSTRAINT pk_Vendors_MerchSales PRIMARY KEY (VendorID, MerchSaleID)
)
```

```
CREATE TABLE Transportation(
    TransportID NUMBER NOT NULL,
    VehicleType VARCHAR(20),
    DriverName VARCHAR(20),
    PickupTime VARCHAR(20),
    DropoffTime VARCHAR(20),
    CONSTRAINT pk_Transportation PRIMARY KEY (TransportID)
)
```

```
CREATE TABLE Artists_Transportation(
    VendorID NUMBER NOT NULL,
    ArtistID NUMBER NOT NULL,
    CONSTRAINT pk_Artists_Transportation PRIMARY KEY (VendorID,ArtistID)
)
```

```
CREATE TABLE Sponsors(
    SponsorID NUMBER NOT NULL,
    Name VARCHAR(20),
    ContactInfo VARCHAR(20),
    CONSTRAINT pk_Sponsors PRIMARY KEY (SponsorID)
)
```

```
CREATE TABLE Sponsorship(
    ArtistSponsorID NUMBER NOT NULL,
    StartDate DATE,
    EndDate DATE,
    SponsorshipAmount NUMBER NOT NULL,
    Notes VARCHAR(20),
    ContractTerms VARCHAR(20),
    SponsorID NUMBER NOT NULL,
    CONSTRAINT pk_Sponsorship PRIMARY KEY (ArtistSponsorID))
```

```
CREATE TABLE Artists_Sponsorship(
    ArtistSponsorID NUMBER NOT NULL,
    ArtistID NUMBER NOT NULL,
    CONSTRAINT pk_Artists_Sponsorship PRIMARY KEY (ArtistSponsorID,ArtistID)
)
```

```

CREATE TABLE Stages(
StageID NUMBER NOT NULL,
Name VARCHAR(20),
Location VARCHAR(20),
Capacity VARCHAR(20),
EquipmentAvailable VARCHAR(20),
CONSTRAINT pk_Stages PRIMARY KEY (StageID)
)

CREATE TABLE Performances(
PerformanceID NUMBER NOT NULL,
PerformanceDate DATE,
StartTime VARCHAR(20),
EndTime VARCHAR(20),
SpecialNotes VARCHAR(20),
ArtistID NUMBER NOT NULL,
StageID NUMBER NOT NULL,
CONSTRAINT pk_Performances PRIMARY KEY (PerformanceID))

CREATE TABLE Staff(
StaffID NUMBER NOT NULL,
Name VARCHAR(20),
Role VARCHAR(20),
ContactInfo VARCHAR(20),
CONSTRAINT pk_Staff PRIMARY KEY (StaffID)
)

CREATE TABLE Performances_Staff(
StaffID NUMBER NOT NULL,
PerformanceID NUMBER NOT NULL,
CONSTRAINT pk_Performances_Staff PRIMARY KEY (PerformanceID,StaffID))

CREATE TABLE SecurityIncidents(
IncidentID NUMBER NOT NULL,
SecurityDate DATE,
Location VARCHAR(20),
Description VARCHAR(20),
Resolution VARCHAR(20),
CONSTRAINT pk_SecurityIncidents PRIMARY KEY (IncidentID)
)

CREATE TABLE Staff_SecurityIncidents(
StaffID NUMBER NOT NULL,
IncidentID NUMBER NOT NULL,

```

```
CONSTRAINT pk_Staff_SecurityIncidents PRIMARY KEY (StaffID,IncidentID)
)
```

```
CREATE TABLE Accommodation(
AccommodationID NUMBER NOT NULL,
HotelName VARCHAR(20),
RoomNumber NUMBER,
CheckinDate DATE,
CheckoutDate DATE,
ArtistID NUMBER NOT NULL,
CONSTRAINT pk_Accommodation PRIMARY KEY (AccommodationID))
```

2. Specifying foreign keys

```
ALTER TABLE Merchandise
ADD CONSTRAINT fk_Merchandise
FOREIGN KEY (ArtistID)
REFERENCES Artists(ArtistID)
```

```
ALTER TABLE Tickets
ADD CONSTRAINT fk_Tickets
FOREIGN KEY (VendorID)
REFERENCES Vendor(VendorID)
```

```
ALTER TABLE VendorSales
ADD CONSTRAINT fk_VendorSales
FOREIGN KEY (VendorID)
REFERENCES Vendor(VendorID)
```

```
ALTER TABLE Sponsorship
ADD CONSTRAINT fk_Sponsorship
FOREIGN KEY (SponsorID)
REFERENCES Sponsors(SponsorID)
```

```
ALTER TABLE Performances
ADD CONSTRAINT fk_Performances
FOREIGN KEY (ArtistID)
REFERENCES Artists(ArtistID)
```

```
ALTER TABLE Performances
```

```
ADD CONSTRAINT fk_Performances2
```

```
FOREIGN KEY (StageID)
```

```
REFERENCES Stages(StageID)
```

```
ALTER TABLE Accommodation
```

```
ADD CONSTRAINT fk_Accommodation
```

```
FOREIGN KEY (ArtistID)
```

```
REFERENCES Artists(ArtistID)
```

```
ALTER TABLE Merchandise_MerchSales
```

```
ADD CONSTRAINT fk_MMS_MerchID
```

```
FOREIGN KEY (MerchID)
```

```
REFERENCES Merchandise(MerchID);
```

```
ALTER TABLE Merchandise_MerchSales
```

```
ADD CONSTRAINT fk_MMS_MerchSaleID
```

```
FOREIGN KEY (MerchSaleID)
```

```
REFERENCES MerchSales(MerchSaleID)
```

```
ALTER TABLE MerchSales_Attendees
```

```
ADD CONSTRAINT fk_MSA_MerchSaleID
```

```
FOREIGN KEY (MerchSaleID)
```

```
REFERENCES MerchSales(MerchSaleID);
```

```
ALTER TABLE MerchSales_Attendees
```

```
ADD CONSTRAINT fk_MSA_AttendeeID
```

```
FOREIGN KEY (AttendeeID)
```

```
REFERENCES Attendees(AttendeeID)
```

```
ALTER TABLE Attendees_Tickets
```

```
ADD CONSTRAINT fk_AT_AttendeeID
```

```
FOREIGN KEY (AttendeeID)
```

```
REFERENCES Attendees(AttendeeID);
```

```
ALTER TABLE Attendees_Tickets
```

```
ADD CONSTRAINT fk_AT_TicketID
```

```
FOREIGN KEY (TicketID)
```

```
REFERENCES Tickets(TicketID);
```

```
ALTER TABLE Vendors_MerchSales
```

```
ADD CONSTRAINT fk_VMS_VendorID
```

```
FOREIGN KEY (VendorID)
```

```
REFERENCES Vendors(VendorID);
```

```
ALTER TABLE Vendors_MerchSales
```

```
ADD CONSTRAINT fk_VMS_MerchSaleID
```

```
FOREIGN KEY (MerchSaleID)
```

```
REFERENCES MerchSales(MerchSaleID);
```

```
ALTER TABLE Artists_Transportation
  ADD CONSTRAINT fk_AT_Artist
  FOREIGN KEY (ArtistID)
    REFERENCES Artists(ArtistID);
```

```
ALTER TABLE Artists_Transportation
  ADD CONSTRAINT fk_AT_Transport
  FOREIGN KEY (TransportID)
    REFERENCES Transportation(TransportID);
```

```
ALTER TABLE Artists_Sponsorship
  ADD CONSTRAINT fk_AS_ArtistID
  FOREIGN KEY (ArtistID)
    REFERENCES Artists(ArtistID);
```

```
ALTER TABLE Artists_Sponsorship
  ADD CONSTRAINT fk_AS_SponsorID
  FOREIGN KEY (ArtistSponsorID)
    REFERENCES Sponsorship(ArtistSponsorID);
```

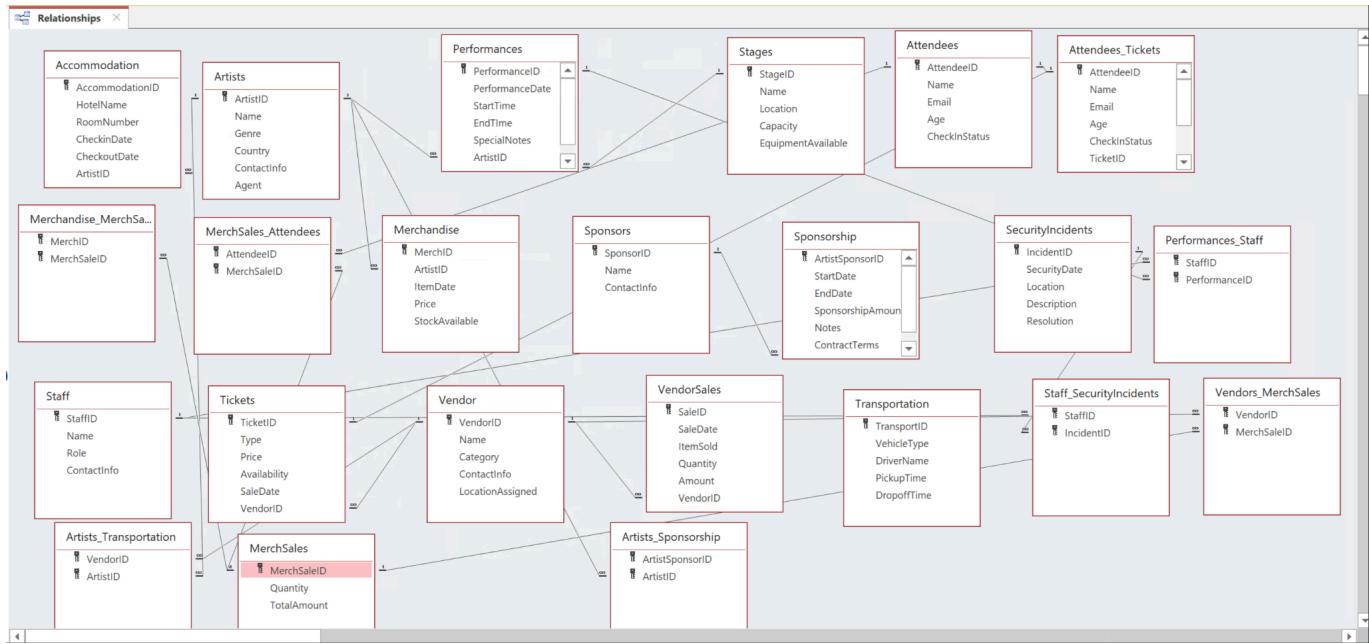
```
ALTER TABLE Performances_Staff
  ADD CONSTRAINT fk_PS_PerformanceID
  FOREIGN KEY (PerformanceID)
    REFERENCES Performances(PerformanceID);
```

```
ALTER TABLE Performances_Staff
  ADD CONSTRAINT fk_PS_StaffID
  FOREIGN KEY (StaffID)
    REFERENCES Staff(StaffID);
```

```
ALTER TABLE Staff_SecurityIncidents
  ADD CONSTRAINT fk_SSI_StaffID
  FOREIGN KEY (StaffID)
    REFERENCES Staff(StaffID);
```

```
ALTER TABLE Staff_SecurityIncidents
  ADD CONSTRAINT fk_SSI_IncidentID
  FOREIGN KEY (IncidentID)
    REFERENCES SecurityIncidents(IncidentID);
```

Relationship View



3. Adding Data to the Tables using SQL INSERT Statements

Artists

```
INSERT INTO Artists VALUES (1, 'Luna', 'Pop', 'USA', 'luna@example.com', 'StarTalent Agency')
```

```
INSERT INTO Artists VALUES (2, 'Kai', 'Rock', 'Japan', 'kai@example.jp', 'Infinity Music')
```

```
INSERT INTO Artists VALUES (3, 'Emma', 'Electronic', 'Canada', 'emma.li@example.ca', 'Northern Sound Mgmt')
```

Merchandise

```
INSERT INTO Merchandise VALUES (1, '#01-10#', 25.99, 120, 1);
```

```
INSERT INTO Merchandise VALUES (2, '#02-05#', 39.50, 80, 2);
```

```
INSERT INTO Merchandise VALUES (3, '#03-12#', 15.00, 200, 3);
```

MerchSales

```
INSERT INTO MerchSales VALUES (1, 2, 51.98);
```

```
INSERT INTO MerchSales VALUES (2, 1, 39.50);
```

```
INSERT INTO MerchSales VALUES (3, 5, 75.00);
```

Merchandise_MerchSales

```
INSERT INTO Merchandise_MerchSales VALUES (1, 1);
INSERT INTO Merchandise_MerchSales VALUES (2, 2);
INSERT INTO Merchandise_MerchSales VALUES (3, 3);
```

Attendees

```
INSERT INTO Attendees VALUES (1, 'Alice', 'alice@example.com', 25, 'CheckedIn');
INSERT INTO Attendees VALUES (2, 'Brian', 'brian@example.com', 30, 'NotCheckedIn');
INSERT INTO Attendees VALUES (3, 'Chloe', 'chloe@example.com', 22, 'CheckedIn');
```

MerchSales_Attendees

```
INSERT INTO MerchSales_Attendees VALUES (1, 1);
INSERT INTO MerchSales_Attendees VALUES (2, 2);
INSERT INTO MerchSales_Attendees VALUES (3, 3);
```

Attendees_Tickets

```
INSERT INTO Attendees_Tickets VALUES (1, 'Alice', 'alice@example.com', 25, 'CheckedIn', 101, 'VIP', 150.00, 'Available', '#01-15#');
INSERT INTO Attendees_Tickets VALUES (2, 'Brian', 'brian@example.com', 30, 'NotCheckedIn', 102, 'General', 85.00, 'SoldOut', '#01-20#');
INSERT INTO Attendees_Tickets VALUES (3, 'Chloe', 'chloe@example.com', 22, 'CheckedIn', 103, 'VIP', 150.00, 'Available', '#01-25#');
```

Tickets

```
INSERT INTO Tickets VALUES (101, 'VIP', 150.00, 'Available', '#01-15#', 1);
INSERT INTO Tickets VALUES (102, 'General', 85.00, 'SoldOut', '#01-20#', 2);
INSERT INTO Tickets VALUES (103, 'VIP', 150.00, 'Available', '#01-25#', 3);
```

Vendors

```
INSERT INTO Vendors VALUES (1, 'FreshBite Foods', 'Snacks', 'contact@freshbite.com', 'Section A');
INSERT INTO Vendors VALUES (2, 'GlowGear', 'Merchandise', 'info@glowgear.com', 'Section B');
INSERT INTO Vendors VALUES (3, 'DrinkWave', 'Beverages', 'support@drinkwave.com', 'Section C');
```

VendorSales

```
INSERT INTO VendorSales VALUES (1, '#01-10#', 'Glow Sticks Pack', 20, 199.80, 2);
INSERT INTO VendorSales VALUES (2, '#01-12#', 'Bottled Water', 50, 125.00, 3);
INSERT INTO VendorSales VALUES (3, '#01-15#', 'Snack Box', 15, 89.25, 1);
```

Vendors_MerchSales

```
INSERT INTO Vendors_MerchSales VALUES (1, 1);
INSERT INTO Vendors_MerchSales VALUES (2, 2);
INSERT INTO Vendors_MerchSales VALUES (3, 3);
```

Transportation

```
INSERT INTO Transportation VALUES (1, 'Van', 'John', '#01-10 14:00#', '#01-10 14:45#');
```

```
INSERT INTO Transportation VALUES (2, 'SUV', 'Angela', #01-11 09:30#, #-01-11 10:15#);
INSERT INTO Transportation VALUES (3, 'Shuttle Bus', 'David', #01-12 16:00#, #01-12
17:10#);
```

Artists_Transportation

```
INSERT INTO Artists_Transportation VALUES (1, 1);
INSERT INTO Artists_Transportation VALUES (2, 2);
INSERT INTO Artists_Transportation VALUES (3, 3);
```

Sponsorship

```
INSERT INTO Sponsorship VALUES (1, #01-01#, #06-01#, 50000.00, 'Annual sponsorship',
'Standard contract terms apply', 1);
INSERT INTO Sponsorship VALUES (2, #02-01#, #07-01#, 75000.00, 'Tour partnership',
'Includes promotional rights', 2);
INSERT INTO Sponsorship VALUES (3, #03-15#, #09-15#, 30000.00, 'Event-specific
sponsor', 'Limited-term contract', 3);
```

Sponsors

```
INSERT INTO Sponsors VALUES (1, 'NovaCorp', 'contact@novacorp.com');
INSERT INTO Sponsors VALUES (2, 'Skyline Media', 'info@skylinemedia.com');
INSERT INTO Sponsors VALUES (3, 'PulseTech', 'support@pulsetech.com');
```

Artists_Sponsorship

```
INSERT INTO Artists_Sponsorship VALUES (1, 1);
INSERT INTO Artists_Sponsorship VALUES (2, 2);
INSERT INTO Artists_Sponsorship VALUES (3, 3);
```

Performances

```
INSERT INTO Performances VALUES (1, #01-10#, '18:00', '20:00', 'Opening act', 1, 1,
120.00, 300, 'Pop');
INSERT INTO Performances VALUES (2, #01-15#, '20:30', '22:00', 'Main event', 2, 2,
200.00, 500, 'Rock');
INSERT INTO Performances VALUES (3, #01-20#, '19:00', '21:00', 'Evening show', 3, 3,
180.00, 400, 'Jazz');
```

Stages

```
INSERT INTO Stages VALUES (1, 'Main Stage', 'North Wing', 1000, 'Lights, Speakers', 'John
', '14:00', #01-05#);
INSERT INTO Stages VALUES (2, 'Side Stage', 'South Wing', 500, 'Basic Sound', 'Jane ',
'12:00', #01-07#);
INSERT INTO Stages VALUES (3, 'Acoustic Stage', 'East Wing', 300, 'Acoustic Setup', 'Mike
', '13:00', #01-09#);
```

Staff

INSERT INTO Staff VALUES (1, 'Alice Brown', 'Usher', 'alice.staff@example.com', '09:00', '17:00', 'Bob Brown');

INSERT INTO Staff VALUES (2, 'Brian Green', 'Technician', 'brian.staff@example.com', '10:00', '18:00', 'Mary Green');

INSERT INTO Staff VALUES (3, 'Chloe White', 'Security', 'chloe.staff@example.com', '08:00', '16:00', 'Tom White');

Performances_Staff

INSERT INTO Performances_Staff VALUES (1, 1, 'Usher', '#01-10 17:30#', '#01-10 20:30#');

INSERT INTO Performances_Staff VALUES (2, 2, 'Sound Tech', '#01-15 19:00#', '#01-15 22:30#');

INSERT INTO Performances_Staff VALUES (3, 3, 'Security', '#01-20 18:30#', '#01-20 21:30#');

SecurityIncidents

INSERT INTO SecurityIncidents VALUES (1, '#01-11#', 'Main Stage', 'Lost wallet reported', 'Resolved', 3, 'Medium', '#01-12#');

INSERT INTO SecurityIncidents VALUES (2, '#01-16#', 'Side Stage', 'Unauthorized entry', 'Resolved', 3, 'High', '#01-17#');

INSERT INTO SecurityIncidents VALUES (3, '#01-21#', 'Acoustic Stage', 'Spilled drink', 'Cleaned', 2, 'Low', '#01-22#');

Staff_SecurityIncidents

INSERT INTO Staff_SecurityIncidents VALUES (1, 1, 'Responding Officer', '17:45', 'Handled quickly');

INSERT INTO Staff_SecurityIncidents VALUES (3, 2, 'Security Lead', '20:15', 'Escorted out');

INSERT INTO Staff_SecurityIncidents VALUES (2, 3, 'Cleaner', '19:30', 'Cleaned area');

Accommodation

INSERT INTO Accommodation VALUES (1, 'Grand Hotel', '101', '#01-09#', '#01-12#', 1, 'GH-202501', 2, 'Near elevator');

INSERT INTO Accommodation VALUES (2, 'City Inn', '202', '#01-14#', '#01-17#', 2, 'CI-202501', 1, 'Late check-in');

INSERT INTO Accommodation VALUES (3, 'Sunset Suites', '303', '#01-19#', '#01-22#', 3, 'SS-202501', 3, 'High floor');

E. Application Implementation

Scenario 1 – Performance Schedule by Artist and Stage

The event manager wants to see the full performance schedule: which artist is playing, at what time, and on which stage.

SQL query

```
SELECT a.Name AS ArtistName, p.PerformanceDate, p.StartTime, p.EndTime,  
s.Name AS StageName, s.Location  
  
FROM (Artists AS a INNER JOIN Performances AS p  
  
ON a.ArtistID = p.ArtistID)  
  
INNER JOIN Stages AS s  
  
ON p.StageID = s.StageID  
  
ORDER BY p.PerformanceDate, p.StartTime;
```

ArtistName	Performance	StartTime	EndTime	StageName	Location
Luna	1/10/2025	18:00	20:00	Main Stage	North Wing
Kai	1/15/2025	20:30	22:00	Side Stage	South Wing
Emma	1/20/2025	19:00	21:00	Acoustic Stage	East Wing

Explanation

This query joins Artists, Performances, and Stages to show a readable performance schedule. Management can quickly see who is performing where and when, which is useful for scheduling and resolving conflicts.

Scenario 2 – Tickets Purchased by Each Attendee

The ticketing team wants to know which attendee bought which type of ticket, and when it was purchased.

SQL query

```

SELECT atd.Name AS AttendeeName, atd.Email, t.Type AS TicketType, t.Price,
t.SaleDate, atd.CheckInStatus

FROM (Attendees AS atd

INNER JOIN Attendees_Tickets AS atk

ON atd.AttendeeID = atk.AttendeeID)

INNER JOIN Tickets AS t

ON atk.TicketID = t.TicketID

ORDER BY t.SaleDate;

```

	AttendeeName	Email	TicketType	Price	SaleDate	CheckInStatus
	Alice	alice@example.com	VIP	\$150.00	1/15/2025	CheckedIn
	Brian	brian@example.com	General	\$85.00	1/20/2025	NotCheckedIn
	Chloe	chloe@example.com	VIP	\$150.00	1/25/2025	CheckedIn

Explanation

This query links Attendees, Attendees_Tickets, and Tickets to show each attendee's ticket type, price, purchase date, and check-in status. It can be used at the help desk to verify purchases or analyze VIP vs General ticket usage.

Scenario 3 – Total Sales by Vendor

The finance team wants to know which vendors generated the most revenue during the festival.

SQL query

```

SELECT v.Name AS VendorName,SUM(vs.Amount) AS TotalSales,
COUNT(vs.SaleID) AS NumberOfTransactions

FROM Vendors AS v

INNER JOIN VendorSales AS vs

ON v.VendorID = vs.VendorID

```

```
GROUP BY v.Name  
ORDER BY TotalSales DESC;
```

VendorName	TotalSales	NumberOfTransactions
GlowGear	\$199.80	1
FreshBite Foods	\$89.25	1
DrinkWave	\$125.00	1

Explanation

This query aggregates VendorSales by vendor, computing total revenue and number of sales per vendor. It helps management decide which vendors to invite back next year and how to negotiate future contracts.

Scenario 4 – Merchandise Revenue by Artist

Business question

The organizers want to see which artists generate the most merchandise revenue.

SQL query

```
SELECT a.Name AS ArtistName, SUM(ms.TotalAmount) AS  
TotalMerchRevenue  
  
FROM (((Artists AS a  
  
INNER JOIN Merchandise AS m  
  
ON a.ArtistID = m.ArtistID)  
  
INNER JOIN Merchandise_MerchSales AS mm  
  
ON m.MerchID = mm.MerchID)
```

```

INNER JOIN MerchSales AS ms
ON mm.MerchSaleID = ms.MerchSaleID)
GROUP BY a.Name
ORDER BY TotalMerchRevenue DESC;

```

ArtistName	TotalMerchRevenue
Luna	\$51.98
Kai	\$39.50
Emma	\$75.00

Explanation

This query joins Artists → Merchandise → Merchandise_MerchSales → MerchSales and sums merchandise revenue per artist. It shows which artists sell the most merch and can support decisions about future booking fees or promotional support.

Scenario 5 – Artist Logistics: Accommodation and Transportation

Business question

The logistics team wants one view that shows where each artist is staying and what transportation is assigned to them.

SQL query

```

SELECT a.Name AS ArtistName,ac.HotelName,
ac.RoomNumber,ac.CheckinDate,ac.CheckoutDate,tr.VehicleType,tr.DriverName
,tr.PickupTime,tr.DropoffTime
FROM (((Artists AS a
LEFT JOIN Accommodation AS ac
ON a.ArtistID = ac.ArtistID)

```

```

LEFT JOIN Artists_Transportation AS at
    ON a.ArtistID = at.ArtistID)
LEFT JOIN Transportation AS tr
    ON at.TransportID = tr.TransportID)
ORDER BY a.Name;

```

ArtistName	HotelName	RoomNumb	CheckinDate	ChecoutDate	VehicleType	DriverName	PickupTime	DropoffTime
Emma	Sunset Suites	303	1/19/2025	1/22/2025	Shuttle Bus	David	16:00	17:10
Kai	City Inn	202	1/14/2025	1/17/2025	SUV	Angela	09:30	10:15
Luna	Grand Hotel	101	1/9/2025	1/12/2025	Van	John	14:00	14:45

Explanation

This query uses LEFT JOINS so artists are shown even if they are missing hotel or transport records. It combines Artists, Accommodation, Artists_Transportation, and Transportation into one report, helping the logistics team avoid mistakes (like forgetting to book a shuttle for an artist).

Scenario 6 – Security Incidents and Responsible Staff

Business question

Security management wants to review each incident and see which staff members handled it.

SQL query

```

SELECT si.IncidentID, si.SecurityDate, si.Location, si.Description, si.Resolution,
st.Name AS StaffName, st.Role
FROM (SecurityIncidents AS si
    INNER JOIN Staff_SecurityIncidents AS ssi
        ON si.IncidentID = ssi.IncidentID)
    INNER JOIN Staff AS st
        ON ssi.StaffID = st.StaffID
ORDER BY si.SecurityDate, si.IncidentID;

```

IncidentID	SecurityDate	Location	Description	Resolution	StaffName	Role
1		Main Stage	Lost wallet reported	Resolved	Alice Brown	Usher
2		Side Stage	Unauthorized entry	Resolved	Chloe White	Security
3		Acoustic Stage	Spilled drink	Cleaned	Brian Green	Technician

Explanation

This query joins SecurityIncidents, Staff_SecurityIncidents, and Staff to show which staff members were involved in resolving each incident. It's useful for accountability, training, and improving security procedures.