ATTENDANCE MANAGEMENT SYSTEM

~ USING MYSQL

OUTLINE

- Database Schema Design
- Storing Records
- Relationships
- Data Entry
- Queries
- Updating Records
- Deleting Records
- Reports and Analysis
- User Interface
- Security

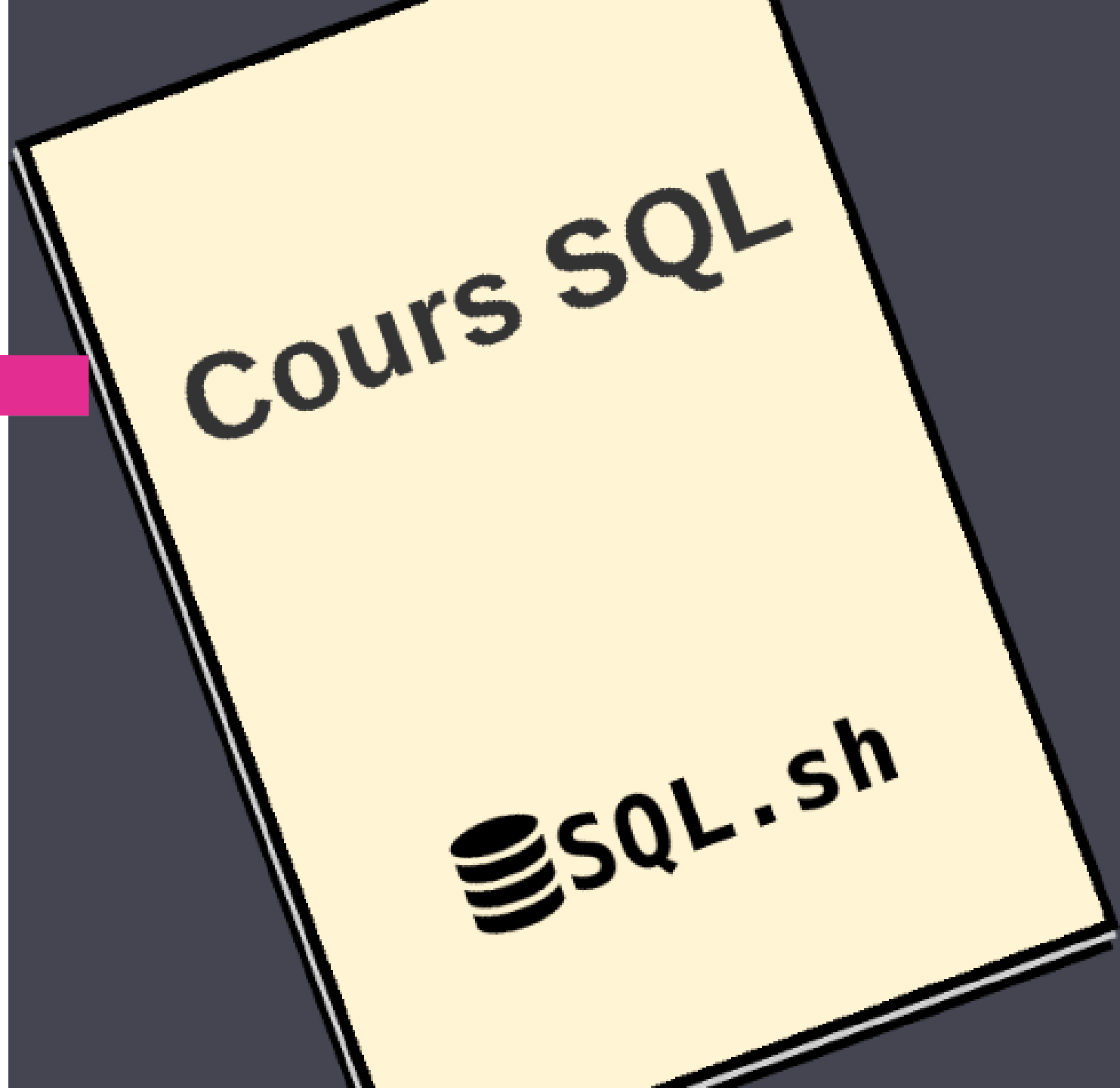
WHAT IS SQL

- **Structured Query Language**
- **Database Management**
- **Data Manipulation**
- **Relational Databases**
- **Standardized Language**

CASE STUDY ON
ATTENDANCE MANAGEMENT SYSTEM

The system tracks

- employees,
- their attendance,
- departments,
- holidays,
- leave requests,
- and work schedules.



NEED FOR ATTENDANCE MANAGEMENT SYSTEM

1. **Accurate Tracking**
2. **Centralized Storage**
3. **Automated Reports**
4. **Easy Data Retrieval:**
5. **Data Integrity**
6. **Scalability**

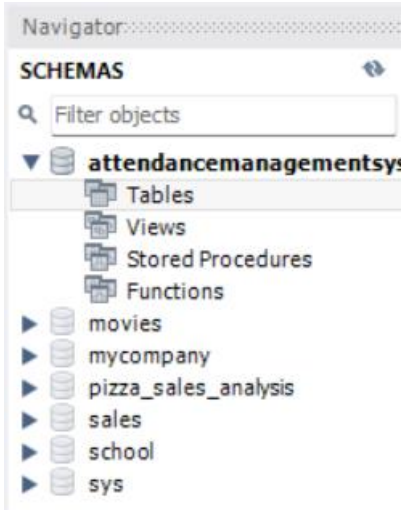
WHAT IS A DATABASE

- **Organized Collection**: Stores structured data systematically.
- **Tables**: Comprised of multiple related tables.
- **Schemas**: Defines the structure of tables and relationships.
- **Queries**: Allows data retrieval and manipulation.
- **Management System**: Operated by a Database Management System (DBMS).

CREATING A DATABASE

```
CREATE DATABASE AttendanceManagementSystem;
use AttendanceManagementSystem;
```

✔	3	17:53:25	CREATE DATABASE AttendanceManagementSystem	1 row(s) affected
✔	5	17:55:32	use AttendanceManagementSystem	0 row(s) affected



WHY WE CREATING TABLES

- **Organize Data**: Structure and categorize data logically.
- **Ensure Data Integrity**: Enforce data types and constraints.
- **Enable Relationships**: Link related data using keys.
- **Efficient Querying**: Optimize data retrieval and manipulation.
- **Manage Permissions**: Control access to specific data.

CREATE TABLE:

WRITE AN SQL STATEMENT TO CREATE ALL TABLES WITH THE SPECIFIED COLUMNS.

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Email VARCHAR(100),
    Phone VARCHAR(15),
    Department VARCHAR(50),
    HireDate DATE
);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
EmployeeID	int	NO	PRI	NULL	
FirstName	varchar(50)	YES		NULL	
LastName	varchar(50)	YES		NULL	
Email	varchar(100)	YES		NULL	
Phone	varchar(15)	YES		NULL	
Department	varchar(50)	YES		NULL	
HireDate	date	YES		NULL	

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Tables_in_attendancemanagementsystem
attendance
departments
employees
holidays
leaverequests
workschedules

CREATE TABLE:

WRITE AN SQL STATEMENT TO CREATE ALL TABLES WITH THE SPECIFIED COLUMNS

```sql
CREATE TABLE Departments (
    DepartmentID INT PRIMARY KEY,
    DepartmentName VARCHAR(50),
    Location VARCHAR(50)
);
desc Departments;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
DepartmentID	int	NO	PRI	NULL	
DepartmentName	varchar(50)	YES		NULL	
Location	varchar(50)	YES		NULL	

```sql
CREATE TABLE Attendance (
    AttendanceID INT PRIMARY KEY,
    EmployeeID INT,
    Date DATE,
    CheckInTime DATETIME,
    CheckOutTime DATETIME,
    Status VARCHAR(10),
    FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
);
desc Attendance;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
AttendanceID	int	NO	PRI	NULL	
EmployeeID	int	YES	MUL	NULL	
Date	date	YES		NULL	
CheckInTime	datetime	YES		NULL	
CheckOutTime	datetime	YES		NULL	
Status	varchar(10)	YES		NULL	

CREATE TABLE:

WRITE AN SQL STATEMENT TO CREATE ALL TABLES WITH THE SPECIFIED COLUMNS

```sql
CREATE TABLE Holidays (
    HolidayID INT PRIMARY KEY,
    HolidayDate DATE,
    HolidayName VARCHAR(50)
);
desc Holidays;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
HolidayID	int	NO	PRI	NULL	
HolidayDate	date	YES		NULL	
HolidayName	varchar(50)	YES		NULL	

```sql
CREATE TABLE LeaveRequests (
    LeaveRequestID INT PRIMARY KEY,
    EmployeeID INT,
    StartDate DATE,
    EndDate DATE,
    LeaveType VARCHAR(20),
    Status VARCHAR(10),
    FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
);
```

```sql
desc LeaveRequests;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

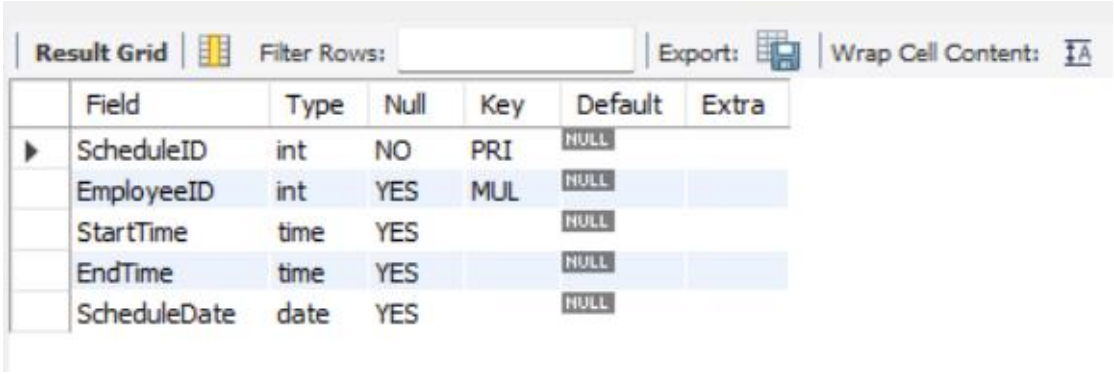
Field	Type	Null	Key	Default	Extra
LeaveRequestID	int	NO	PRI	NULL	
EmployeeID	int	YES	MUL	NULL	
StartDate	date	YES		NULL	
EndDate	date	YES		NULL	
LeaveType	varchar(20)	YES		NULL	
Status	varchar(10)	YES		NULL	

CREATE TABLE:

WRITE AN SQL STATEMENT TO CREATE ALL TABLES WITH THE SPECIFIED COLUMNS

```
CREATE TABLE WorkSchedules (
    ScheduleID INT PRIMARY KEY,
    EmployeeID INT,
    StartTime TIME,
    EndTime TIME,
    ScheduleDate DATE,
    FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
);
```

```
desc WorkSchedules;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
ScheduleID	int	NO	PRI	NULL	
EmployeeID	int	YES	MUL	NULL	
StartTime	time	YES		NULL	
EndTime	time	YES		NULL	
ScheduleDate	date	YES		NULL	

WHY WE NEED TO INSERT RECORDS

- **Populate Data**: Fill tables with relevant data for analysis and operations.
- **Enable Queries**: Provide data that can be retrieved, filtered, and analyzed.
- **Maintain Records**: Keep accurate and up-to-date records for tracking and reporting.
- **Support Applications**: Supply backend data for applications to function correctly.
- **Ensure Functionality**: Allow database functionalities like updates, deletions, and relationships to operate on real data.

INSERT RECORDS:

INSERT AT LEAST 10 RECORDS IN ALL THE TABLES.

```
55    -- Insert records into Employees
56
57 •  INSERT INTO Employees VALUES
58    (1, 'Shreyas', 'Jadhav', 'Shresh.j@itvedant.com', '9832148761', 'HR', '2021-05-15'),
59    (2, 'Nishtha', 'Yadav', 'Nish.y@mac.com', '0986354271', 'Sales', '2019-11-30'),
70    (3, 'Ayush', 'Yadav', 'Ay.yadav@Laren.com', '126198834', 'IT', '2020-01-15'),
71    (4, 'Sakshi', 'Jadhav', 'Sakshi@work.com', '6313894721', 'Finance', '2018-07-23'),
72    (5, 'Yamini', 'Bhapatre', 'yami.b@workit.com', '9517539563', 'HR', '2022-02-14'),
73    (6, 'Gaurang', 'Unde', 'G.unde@testit.com', '819237465', 'Sales', '2023-01-01'),
74    (7, 'Pranjal', 'Sawalekar', 'Pranjal.p@it.com', '1591591590', 'IT', '2017-10-10'),
75    (8, 'Rishi', 'Sawant', 'Rishi.sawant@cowork.com', '7537537530', 'Finance', '2021-03-12'),
76    (9, 'Aaditya', 'Negi', 'Aadi.Negi@work.com', '9519519510', 'HR', '2022-06-18'),
77    (10, 'Rushda', 'Shaikh', 'Rushda.shaikh@coworkit.com', '3573573570', 'IT', '2016-05-05');
78 •  select * from Employees;
79
```

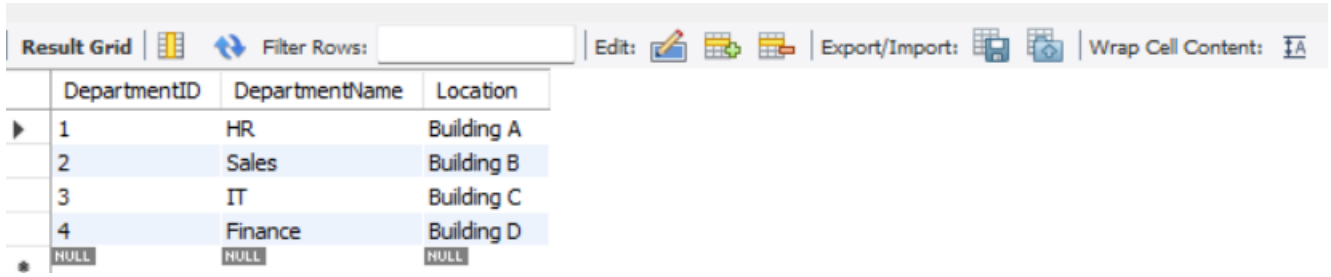
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

EmployeeID	FirstName	LastName	Email	Phone	Department	HireDate
1	Shreyas	Jadhav	Shresh.j@itvedant.com	9832148761	HR	2021-05-15
2	Nishtha	Yadav	Nish.y@mac.com	0986354271	Sales	2019-11-30
3	Ayush	Yadav	Ay.yadav@Laren.com	126198834	IT	2020-01-15
4	Sakshi	Jadhav	Sakshi@work.com	6313894721	Finance	2018-07-23
5	Yamini	Bhapatre	yami.b@workit.com	9517539563	HR	2022-02-14
6	Gaurang	Unde	G.unde@testit.com	819237465	Sales	2023-01-01
7	Pranjal	Sawalekar	Pranjal.p@it.com	1591591590	IT	2017-10-10
8	Rishi	Sawant	Rishi.sawant@cowork.com	7537537530	Finance	2021-03-12
9	Aaditya	Negi	Aadi.Negi@work.com	9519519510	HR	2022-06-18
10	Rushda	Shaikh	Rushda.shaikh@coworkit.com	3573573570	IT	2016-05-05
NULL	NULL	NULL	NULL	NULL	NULL	NULL

INSERT RECORDS:

INSERT AT LEAST 10 RECORDS IN ALL THE TABLES.

```
-- Insert records into Departments
INSERT INTO Departments VALUES
(1, 'HR', 'Building A'),
(2, 'Sales', 'Building B'),
(3, 'IT', 'Building C'),
(4, 'Finance', 'Building D');
select * from Departments;
```



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

DepartmentID	DepartmentName	Location
1	HR	Building A
2	Sales	Building B
3	IT	Building C
4	Finance	Building D
NULL	NULL	NULL

INSERT RECORDS:

INSERT AT LEAST 10 RECORDS IN ALL THE TABLES.

```sql
-- Insert records into Attendance
INSERT INTO Attendance VALUES
(1, 1, '2024-07-01', '2024-07-01 09:00:00', '2024-07-01 17:00:00', 'Present'),
(2, 2, '2024-07-01', '2024-07-01 09:15:00', '2024-07-01 17:15:00', 'Late'),
(3, 3, '2024-07-01', '2024-07-01 09:00:00', '2024-07-01 17:00:00', 'Present'),
(4, 4, '2024-07-01', '2024-07-01 09:00:00', '2024-07-01 17:00:00', 'Present'),
(5, 5, '2024-07-01', '2024-07-01 09:00:00', '2024-07-01 17:00:00', 'Present'),
(6, 6, '2024-07-01', '2024-07-01 09:40:00', '2024-07-01 17:40:00', 'Late'),
(7, 7, '2024-07-01', '2024-07-01 09:00:00', '2024-07-01 17:00:00', 'Present'),
(8, 8, '2024-07-01', '2024-07-01 09:00:00', '2024-07-01 17:00:00', 'Present'),
(9, 9, '2024-07-01', '2024-07-01 09:15:00', '2024-07-01 17:15:00', 'Late'),
(10, 10, '2024-07-01', '2024-07-01 09:00:00', '2024-07-01 17:00:00', 'Present');
select * from Attendance;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

AttendanceID	EmployeeID	Date	CheckInTime	CheckOutTime	Status
1	1	2024-07-01	2024-07-01 09:00:00	2024-07-01 17:00:00	Present
2	2	2024-07-01	2024-07-01 09:15:00	2024-07-01 17:15:00	Late
3	3	2024-07-01	2024-07-01 09:00:00	2024-07-01 17:00:00	Present
4	4	2024-07-01	2024-07-01 09:00:00	2024-07-01 17:00:00	Present
5	5	2024-07-01	2024-07-01 09:00:00	2024-07-01 17:00:00	Present
6	6	2024-07-01	2024-07-01 09:40:00	2024-07-01 17:40:00	Late
7	7	2024-07-01	2024-07-01 09:00:00	2024-07-01 17:00:00	Present
8	8	2024-07-01	2024-07-01 09:00:00	2024-07-01 17:00:00	Present
9	9	2024-07-01	2024-07-01 09:15:00	2024-07-01 17:15:00	Late
10	10	2024-07-01	2024-07-01 09:00:00	2024-07-01 17:00:00	Present
NULL	NULL	NULL	NULL	NULL	NULL

INSERT RECORDS:

INSERT AT LEAST 10 RECORDS IN ALL THE TABLES.

```
-- Insert records into Holidays
INSERT INTO Holidays VALUES
(1, '2024-01-01', 'New Year'),
(2, '2024-08-15', 'Independence Day'),
(3, '2024-12-25', 'Christmas'),
(4, '2024-11-28', 'Thanksgiving'),
(5, '2024-05-27', 'Memorial Day'),
(6, '2024-09-02', 'Labor Day'),
(7, '2024-11-11', 'Veterans Day'),
(8, '2024-10-31', 'Halloween'),
(9, '2024-02-14', 'Valentine's Day'),
(10, '2024-04-01', 'April Fool's Day');
select * from Holidays;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

HolidayID	HolidayDate	HolidayName
1	2024-01-01	New Year
2	2024-08-15	Independence Day
3	2024-12-25	Christmas
4	2024-11-28	Thanksgiving
5	2024-05-27	Memorial Day
6	2024-09-02	Labor Day
7	2024-11-11	Veterans Day
8	2024-10-31	Halloween
9	2024-02-14	Valentine's Day
10	2024-04-01	April Fool's Day
NULL	NULL	NULL

INSERT RECORDS:

INSERT AT LEAST 10 RECORDS IN ALL THE TABLES

```
INSERT INTO LeaveRequests VALUES
(1, 1, '2024-07-01', '2024-07-05', 'Vacation', 'Approved'),
(2, 2, '2024-07-10', '2024-07-15', 'Sick', 'Pending'),
(3, 3, '2024-07-20', '2024-07-25', 'Sick', 'Rejected'),
(4, 4, '2024-08-01', '2024-08-10', 'Vacation', 'Approved'),
(5, 5, '2024-08-15', '2024-08-20', 'Sick', 'Approved'),
(6, 6, '2024-09-01', '2024-09-05', 'Vacation', 'Pending'),
(7, 7, '2024-09-10', '2024-09-15', 'Sick', 'Rejected'),
(8, 8, '2024-10-01', '2024-10-05', 'Vacation', 'Approved'),
(9, 9, '2024-10-10', '2024-10-15', 'Sick', 'Approved'),
(10, 10, '2024-11-01', '2024-11-05', 'Vacation', 'Pending');
select * from LeaveRequests;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

LeaveRequestID	EmployeeID	StartDate	EndDate	LeaveType	Status
1	1	2024-07-01	2024-07-05	Vacation	Approved
2	2	2024-07-10	2024-07-15	Sick	Pending
3	3	2024-07-20	2024-07-25	Sick	Rejected
4	4	2024-08-01	2024-08-10	Vacation	Approved
5	5	2024-08-15	2024-08-20	Sick	Approved
6	6	2024-09-01	2024-09-05	Vacation	Pending
7	7	2024-09-10	2024-09-15	Sick	Rejected
8	8	2024-10-01	2024-10-05	Vacation	Approved
9	9	2024-10-10	2024-10-15	Sick	Approved
10	10	2024-11-01	2024-11-05	Vacation	Pending
NULL	NULL	NULL	NULL	NULL	NULL

INSERT RECORDS:

INSERT AT LEAST 10 RECORDS IN ALL THE TABLES

```
-- Insert records into WorkSchedules
INSERT INTO WorkSchedules VALUES
(1, 1, '09:00:00', '17:00:00', '2024-07-01'),
(2, 2, '09:00:00', '17:00:00', '2024-07-01'),
(3, 3, '09:00:00', '17:00:00', '2024-07-01'),
(4, 4, '09:00:00', '17:00:00', '2024-07-01'),
(5, 5, '09:00:00', '17:00:00', '2024-07-01'),
(6, 6, '09:00:00', '17:00:00', '2024-07-01'),
(7, 7, '09:00:00', '17:00:00', '2024-07-01'),
(8, 8, '09:00:00', '17:00:00', '2024-07-01'),
(9, 9, '09:00:00', '17:00:00', '2024-07-01'),
(10, 10, '09:00:00', '17:00:00', '2024-07-01');
select * from WorkSchedules;
```

ScheduleID	EmployeeID	StartTime	EndTime	ScheduleDate
1	1	09:00:00	17:00:00	2024-07-01
2	2	09:00:00	17:00:00	2024-07-01
3	3	09:00:00	17:00:00	2024-07-01
4	4	09:00:00	17:00:00	2024-07-01
5	5	09:00:00	17:00:00	2024-07-01
6	6	09:00:00	17:00:00	2024-07-01
7	7	09:00:00	17:00:00	2024-07-01
8	8	09:00:00	17:00:00	2024-07-01
9	9	09:00:00	17:00:00	2024-07-01
10	10	09:00:00	17:00:00	2024-07-01
NULL	NULL	NULL	NULL	NULL

SELECT RECORDS:

WRITE A QUERY TO SELECT ALL ATTENDANCE RECORDS FROM THE ATTENDANCE TABLE WHERE THE STATUS IS 'LATE'.

```
-- Select all attendance records where Status is 'Late'
SELECT * FROM Attendance WHERE Status = 'Late';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

AttendanceID	EmployeeID	Date	CheckInTime	CheckOutTime	Status
2	2	2024-07-01	2024-07-01 09:15:00	2024-07-01 17:15:00	Late
6	6	2024-07-01	2024-07-01 09:40:00	2024-07-01 17:40:00	Late
9	9	2024-07-01	2024-07-01 09:15:00	2024-07-01 17:15:00	Late
NULL	NULL	NULL	NULL	NULL	NULL

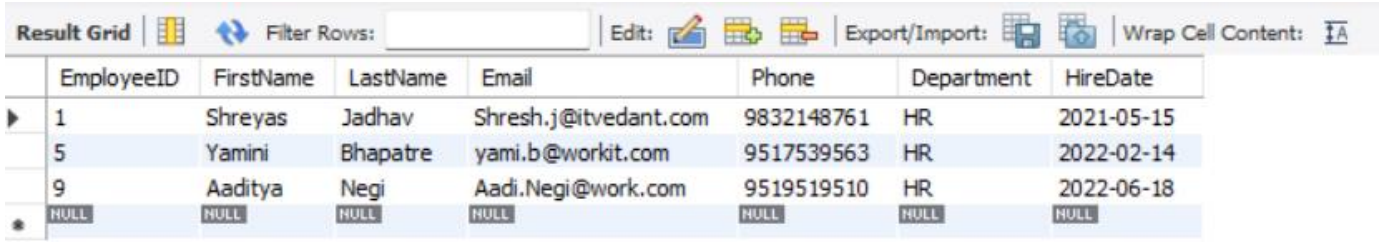
SELECT

- This query retrieves all columns for records in the Attendance table where the Status column is 'Late'.
  '*' is a wildcard used in a select statement to indicate that all the column from table (attendance) should be retrieved.
  'Where' here is used to to filter the records based on specified condition ie (late), selecting those rows will meet the criteria.

WHERE CLAUSE (AND/OR):

WRITE A QUERY TO SELECT ALL EMPLOYEES FROM THE EMPLOYEES TABLE WHO WORK IN THE 'HR' DEPARTMENT AND WERE HIRED AFTER JANUARY 1, 2020.

```sql
-- Select all employees who work in the 'HR' department and were hired after January 1, 2020
SELECT * FROM Employees
WHERE Department = 'HR'
AND HireDate > '2020-01-01';
```



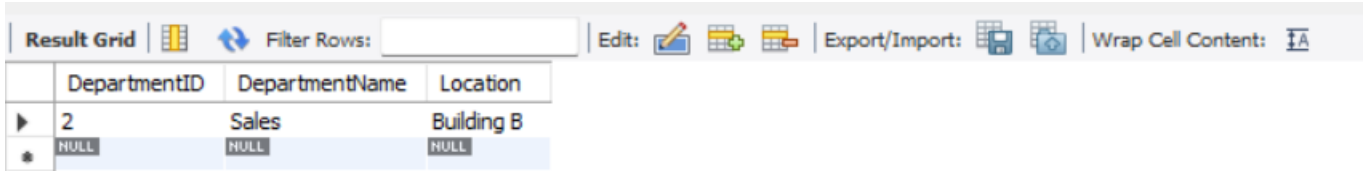
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

EmployeeID	FirstName	LastName	Email	Phone	Department	HireDate
1	Shreyas	Jadhav	Shresh.j@itvedant.com	9832148761	HR	2021-05-15
5	Yamini	Bhapatre	yami.b@workit.com	9517539563	HR	2022-02-14
9	Aaditya	Negi	Aadi.Negi@work.com	9519519510	HR	2022-06-18
NULL	NULL	NULL	NULL	NULL	NULL	NULL

LIKE OPERATOR:

WRITE A QUERY TO SELECT ALL DEPARTMENTS WHERE THE DEPARTMENTNAME CONTAINS 'SALES'.

```
-- Select all departments where the DepartmentName contains 'Sales'
SELECT * FROM Departments
WHERE DepartmentName LIKE '%Sales%';
```



DepartmentID	DepartmentName	Location
2	Sales	Building B
NULL	NULL	NULL

LIKE OPERATOR

The LIKE operator in SQL is used to search for a specified pattern in a column.
It allows for flexible matching using wildcards:

- % matches any sequence of characters.
- _ matches a single character.

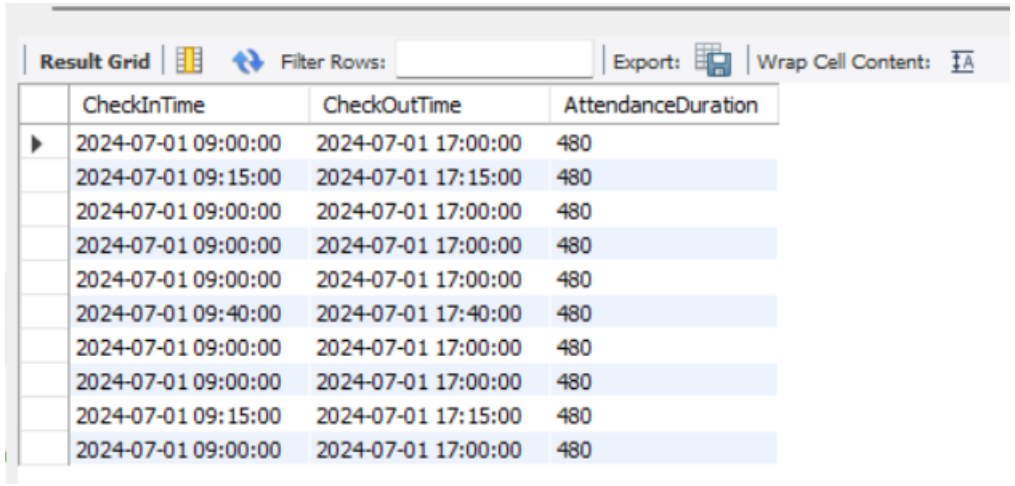
This query retrieves records from the `Departments` table where `DepartmentName` includes the substring 'Sales'. `%` is a wildcard representing any sequence of characters.

CASE STATEMENT:

WRITE A QUERY TO SELECT CHECKINTIME, CHECKOUTTIME, AND A NEW COLUMN ATTENDANCEDURATION FROM THE ATTENDANCE TABLE. CALCULATE ATTENDANCEDURATION AS THE DIFFERENCE BETWEEN CHECKOUTTIME AND

CHECKINTIME.

```sql
-- Select CheckInTime, CheckOutTime, and calculate AttendanceDuration
SELECT
    CheckInTime,
    CheckOutTime,
    TIMESTAMPDIFF(MINUTE, CheckInTime, CheckOutTime) AS AttendanceDuration
FROM Attendance;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CheckInTime	CheckOutTime	AttendanceDuration
2024-07-01 09:00:00	2024-07-01 17:00:00	480
2024-07-01 09:15:00	2024-07-01 17:15:00	480
2024-07-01 09:00:00	2024-07-01 17:00:00	480
2024-07-01 09:00:00	2024-07-01 17:00:00	480
2024-07-01 09:00:00	2024-07-01 17:00:00	480
2024-07-01 09:40:00	2024-07-01 17:40:00	480
2024-07-01 09:00:00	2024-07-01 17:00:00	480
2024-07-01 09:00:00	2024-07-01 17:00:00	480
2024-07-01 09:15:00	2024-07-01 17:15:00	480
2024-07-01 09:00:00	2024-07-01 17:00:00	480

CASE STATEMENT

CheckInTime and CheckOutTime: These are the columns from the Attendance table.

This function calculates the difference between CheckOutTime and CheckInTime in minutes.

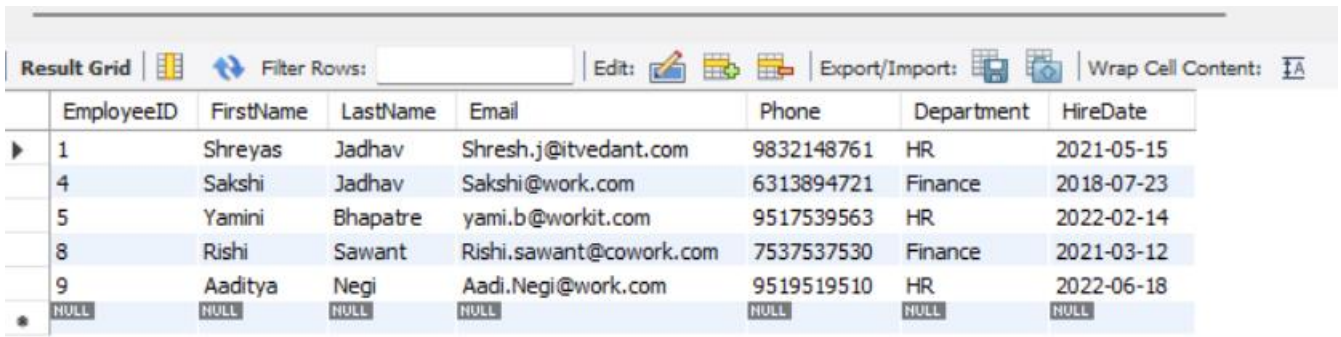
You can change the first argument to second, HOUR, or other units depending on the required format for AttendanceDuration.

Calculate the duration: We use the TIMESTAMPDIFF function or the subtraction method to calculate the duration between CheckOutTime and CheckInTime

SUBQUERY:

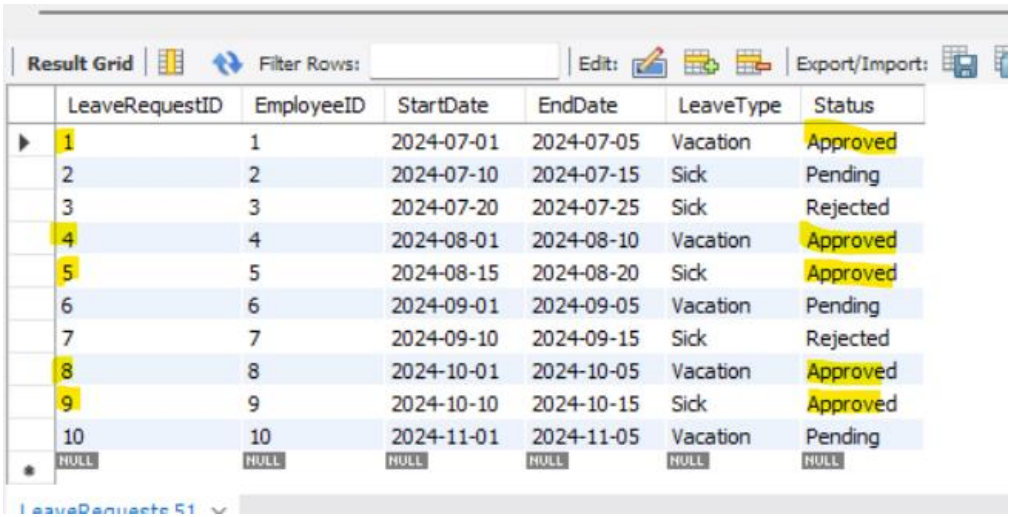
WRITE A QUERY TO FIND ALL EMPLOYEES WHO HAVE AT LEAST ONE 'APPROVED' LEAVE REQUEST. USE A SUBQUERY IN THE WHERE CLAUSE TO FIND THESE EMPLOYEEIDS.

```
-- Find all employees who have at least one 'Approved' leave request
SELECT * FROM Employees
WHERE EmployeeID IN (SELECT EmployeeID
                     FROM LeaveRequests
                     WHERE Status = 'Approved');
```



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

EmployeeID	FirstName	LastName	Email	Phone	Department	HireDate
1	Shreyas	Jadhav	Shresh.j@itvedant.com	9832148761	HR	2021-05-15
4	Sakshi	Jadhav	Sakshi@work.com	6313894721	Finance	2018-07-23
5	Yamini	Bhapatre	yami.b@workit.com	9517539563	HR	2022-02-14
8	Rishi	Sawant	Rishi.sawant@cowork.com	7537537530	Finance	2021-03-12
9	Aaditya	Negi	Aadi.Negi@work.com	9519519510	HR	2022-06-18
NULL	NULL	NULL	NULL	NULL	NULL	NULL



Result Grid | Filter Rows: | Edit: | Export/Import:

LeaveRequestID	EmployeeID	StartDate	EndDate	LeaveType	Status
1	1	2024-07-01	2024-07-05	Vacation	Approved
2	2	2024-07-10	2024-07-15	Sick	Pending
3	3	2024-07-20	2024-07-25	Sick	Rejected
4	4	2024-08-01	2024-08-10	Vacation	Approved
5	5	2024-08-15	2024-08-20	Sick	Approved
6	6	2024-09-01	2024-09-05	Vacation	Pending
7	7	2024-09-10	2024-09-15	Sick	Rejected
8	8	2024-10-01	2024-10-05	Vacation	Approved
9	9	2024-10-10	2024-10-15	Sick	Approved
10	10	2024-11-01	2024-11-05	Vacation	Pending
NULL	NULL	NULL	NULL	NULL	NULL

The subquery returns a list of EmployeeIDs who have at least one 'Approved' leave request.

SUBQUERY

A SUBQUERY IS A QUERY WITHIN A QUERY.

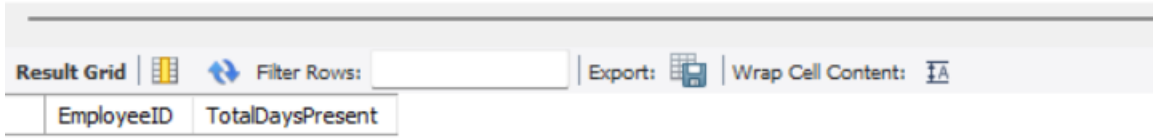
Main Query:
SELECT EmployeeID, EmployeeName FROM Employees: Selects the EmployeeID and EmployeeName columns from the Employees table

The subquery returns a list of EmployeeIDs who have at least one 'Approved' leave request

GROUP BY:

WRITE A QUERY TO GET THE TOTAL NUMBER OF DAYS EACH EMPLOYEE WAS PRESENT IN THE CURRENT MONTH. GROUP THE RESULTS BY EMPLOYEEID.

```sql
-- Get the total number of days each employee was present in the current month
SELECT EmployeeID, COUNT(*) AS TotalDaysPresent
FROM Attendance
WHERE Status = 'Present'
AND MONTH(Date) = MONTH(CURRENT_DATE())
AND YEAR(Date) = YEAR(CURRENT_DATE())
GROUP BY EmployeeID;
```



GROUP BY

- GROUP BY: This clause groups the rows that have the same values in specified columns into aggregated data. In this case, it groups the rows by EmployeeID
  This query retrieves the EmployeeID and the total number of days (TotalDaysPresent) that each employee was present in the current month and year.

  It filters the data to include only those rows where the Status is 'Present' and groups the results by EmployeeID to provide a count of the present days for each employee within the specified time frame.

HAVING CLAUSE:

WRITE A QUERY TO GET THE TOTAL NUMBER OF LEAVE REQUESTS FOR EACH EMPLOYEE, BUT ONLY INCLUDE EMPLOYEES WITH MORE THAN 3 LEAVE REQUESTS. USE THE HAVING CLAUSE.

```sql
-- Get the total number of leave requests for each employee, only include employees with more than 3 leave requests
SELECT EmployeeID, COUNT(*) AS TotalLeaveRequests
FROM LeaveRequests
GROUP BY EmployeeID
HAVING COUNT(*) > 3;
```

HAVING CLAUSE

HAVING: This clause is used to filter the groups created by the GROUP BY clause based on a specified condition.

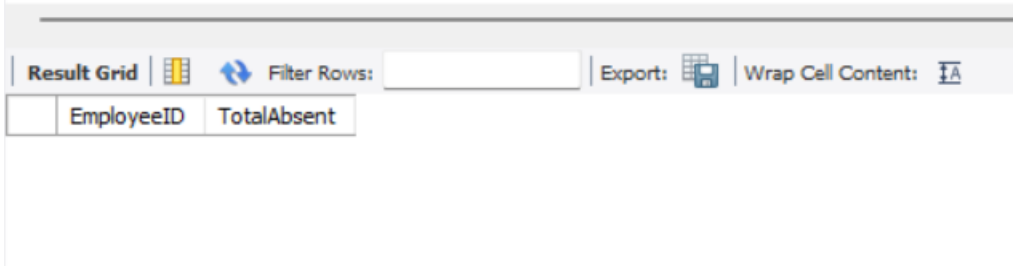
This query retrieves the EmployeeID and the total number of leave requests (TotalLeaveRequests) for each employee from the LeaveRequests table. It groups the data by EmployeeID and includes only those groups where the number of leave requests is greater than 3.

The HAVING clause is used to apply this filter condition on the grouped data.

LIMIT:

WRITE A QUERY TO SELECT THE TOP 5 EMPLOYEES WITH THE MOST NUMBER OF 'ABSENT' STATUSES IN THE PAST YEAR.

```sql
-- Select the top 5 employees with the most number of 'Absent' statuses in the past year
SELECT EmployeeID, COUNT(*) AS TotalAbsent
FROM Attendance
WHERE Status = 'Absent'
AND Date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
GROUP BY EmployeeID
ORDER BY TotalAbsent DESC
LIMIT 5;
```



LIMIT

RESTRICTS THE NUMBER OF ROWS RETURNED BY THE QUERY

This query retrieves the EmployeeID and the total number of absences (TotalAbsences) for each employee from the Attendance table, but only for absences in the past year.
It groups the data by EmployeeID, orders the results by the total number of absences in descending order, and limits the result to the top 5 employees with the most absences.

INNER JOIN:

WRITE A QUERY TO JOIN EMPLOYEES WITH DEPARTMENTS TO GET A LIST OF ALL EMPLOYEES WITH THEIR DEPARTMENT NAMES.

```
200
201     -- Join Employees with Departments to get a list of all employees with their department names
202 •   SELECT e.EmployeeID, e.FirstName, e.LastName, d.DepartmentName
203     FROM Employees e
204     INNER JOIN Departments d ON e.Department = d.DepartmentName;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EmployeeID	FirstName	LastName	DepartmentName
1	Shreyas	Jadhav	HR
2	Nishtha	Yadav	Sales
3	Ayush	Yadav	IT
4	Sakshi	Jadhav	Finance
5	Yamini	Bhapatre	HR
6	Gaurang	Unde	Sales
7	Pranjal	Sawalekar	IT
8	Rishi	Sawant	Finance
9	Aaditya	Negi	HR
10	Rushda	Shaikh	IT

INNER JOIN

- INNER JOIN: Specifies the type of join operation to combine rows from both tables where there is a match in the joining columns.

This query joins the Employees table with the Departments table on the DepartmentID column, retrieving the EmployeeID and EmployeeName from the Employees table and the DepartmentName from the Departments table. The result is a list of all employees along with their respective department names.

The INNER JOIN ensures that only employees who have a matching department in the Departments table are included in the result set.

OUTER JOIN:

WRITE A QUERY TO GET A LIST OF ALL EMPLOYEES AND ANY ASSOCIATED LEAVEREQUESTS. INCLUDE EMPLOYEES WHO MIGHT NOT HAVE ANY LEAVE REQUESTS.

```
206    -- Get a list of all employees and any associated LeaveRequests
207 •  SELECT e.EmployeeID, e.FirstName, e.LastName, lr.LeaveRequestID, lr.StartDate, lr.EndDate, lr.LeaveType, lr.Status
208    FROM Employees e
209    LEFT JOIN LeaveRequests lr ON e.EmployeeID = lr.EmployeeID;
210
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EmployeeID	FirstName	LastName	LeaveRequestID	StartDate	EndDate	LeaveType	Status
1	Shreyas	Jadhav	1	2024-07-01	2024-07-05	Vacation	Approved
2	Nishtha	Yadav	2	2024-07-10	2024-07-15	Sick	Pending
3	Ayush	Yadav	3	2024-07-20	2024-07-25	Sick	Rejected
4	Sakshi	Jadhav	4	2024-08-01	2024-08-10	Vacation	Approved
5	Yamini	Bhapatre	5	2024-08-15	2024-08-20	Sick	Approved
6	Gaurang	Unde	6	2024-09-01	2024-09-05	Vacation	Pending
7	Pranjal	Sawalekar	7	2024-09-10	2024-09-15	Sick	Rejected
8	Rishi	Sawant	8	2024-10-01	2024-10-05	Vacation	Approved
9	Aaditya	Negi	9	2024-10-10	2024-10-15	Sick	Approved
10	Rushda	Shaikh	10	2024-11-01	2024-11-05	Vacation	Pending

OUTER JOIN

This query retrieves a list of all employees and their associated leave requests.

By using a LEFT OUTER JOIN, it ensures that all employees are included in the result set, even if they have no leave requests.

The columns selected include EmployeeID and EmployeeName from the Employees table, and LeaveRequestID, LeaveDate, and Status from the LeaveRequests table.

If an employee has no leave requests, the columns from the LeaveRequests table will have NULL values.

JOIN WITH AGGREGATION:

WRITE A QUERY TO GET THE AVERAGE NUMBER OF HOURS WORKED PER DAY FOR EACH EMPLOYEE. USE AN INNER JOIN BETWEEN WORKSCHEDULES AND ATTENDANCE, AND GROUP BY EMPLOYEEID.

```
212    -- Get the average number of hours worked per day for each employee
213 •  SELECT
214        ws.EmployeeID,
215        AVG(TIMESTAMPDIFF(HOUR, a.CheckInTime, a.CheckOutTime)) AS AvgHoursWorkedPerDay
216    FROM WorkSchedules ws
217    INNER JOIN Attendance a ON ws.EmployeeID = a.EmployeeID AND ws.ScheduleDate = a.Date
218    GROUP BY ws.EmployeeID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EmployeeID	AvgHoursWorkedPerDay
1	8.0000
2	8.0000
3	8.0000
4	8.0000
5	8.0000
6	8.0000
7	8.0000
8	8.0000
9	8.0000
10	8.0000

JOIN WITH AGGREGATION

This query retrieves the average number of hours worked per day for each employee.

It joins the WorkSchedules and Attendance tables on EmployeeID and WorkDate, calculates the difference in hours between CheckInTime and CheckOutTime, and then computes the average of these hours for each employee using the AVG function.

The results are grouped by EmployeeID to get the average hours worked per day for each employee.

SUBQUERY WITH JOIN:

WRITE A QUERY TO FIND ALL EMPLOYEES WHO HAVE WORKED ON DAYS THAT ARE HOLIDAYS. USE A SUBQUERY TO FILTER DATES THAT ARE IN THE HOLIDAYS TABLE.

```
220    -- Find all employees who have worked on days that are holidays
221 •  SELECT e.EmployeeID, e.FirstName, e.LastName
222    FROM Employees e
223    WHERE EXISTS (SELECT 1
224                  FROM Attendance a
225                  JOIN Holidays h ON a.Date = h.HolidayDate
226                  WHERE a.EmployeeID = e.EmployeeID);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

EmployeeID	FirstName	LastName
NULL	NULL	NULL

SUBQUERY WITH JOINS

This query retrieves a list of all employees who have worked on days that are holidays.

It joins the Employees table with the Attendance table on EmployeeID, and uses a subquery to filter WorkDate values that match HolidayDate values from the Holidays table.

The result includes EmployeeID, EmployeeName, and the corresponding WorkDate when the employee worked on a holiday.

ADVANCED JOIN:

WRITE A QUERY TO LIST FIRSTNAME, LASTNAME, DEPARTMENTNAME, AND HOLIDAYNAME FOR ALL EMPLOYEES WHO HAVE THEIR CHECKINTIME ON A HOLIDAY. USE INNER JOIN AND LEFT JOIN AS NECESSARY

TO GET ALL REQUIRED DETAILS.

```
229    -- List FirstName, LastName, DepartmentName, and HolidayName for all employees who have their CheckInTime on a holiday
230 •  SELECT
231        e.FirstName,
232        e.LastName,
233        d.DepartmentName,
234        h.HolidayName
235    FROM Employees e
236    INNER JOIN Attendance a ON e.EmployeeID = a.EmployeeID
237    INNER JOIN Departments d ON e.Department = d.DepartmentName
238    LEFT JOIN Holidays h ON DATE(a.CheckInTime) = h.HolidayDate
239    WHERE h.HolidayDate IS NOT NULL;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

FirstName	LastName	DepartmentName	HolidayName

ADVANCED JOIN

This query retrieves the FirstName, LastName, DepartmentName, and HolidayName for all employees who have their CheckInTime on a holiday.

It joins the Employees table with the Attendance table on EmployeeID, then joins with the Holidays table on the date part of CheckInTime matching HolidayDate.

Finally, it uses a LEFT JOIN to include the DepartmentName from the Departments table, ensuring that employees without a department are still included.