



REMOTE WAREHOUSE MANAGEMENT

Contents

1. Problem Description	4
1.1. Problem Statement	4
1.2. Organization Description and goals	4
1.3. Business requirements	6
1.4. Relational Data Model	6
1.5. Assumptions about Data Entities and Relationships	7
1.6. Scenarios covered for Database	7
2. Entity Relationship Diagram	8
3. Scope of the database	9
3.1.1 Employee Table	9
3.1.2 Department Table	9
3.1.3 Manager Table	9
3.1.4 WarehouseTable	10
3.1.5 Section Table	10
3.1.6 Product Table	10
3.1.7 Projects Table	10
3.1.8 Log detail Table	11
3.1.9 Remote Desk Table	11
3.1.10 Remote Desk Warning Table	11
3.1.11 Meeting Table	11
3.1.12 Meeting Attendance Table	12
3.1.13 Project Team Table	12
3.1.14 Section Codes Table	12
3.2 3NF DATABASE	13
4. Retrieving the data according to business goals	14
4.1. Query 1	14
4.2. Query 2 (Stored Function)	14
4.3. Query 3	15
4.4. Query 4 (Views)	15
4.5. Query 5	16
4.6. Query 6	17
4.7. Query 7	17
4.8. Query 8	17
4.9. Query 9 (Trigger)	18
4.10. Query 10 (Stored Function & Stored Procedure)	18
4.11 Query 11	19
5. Members contribution	21

1. Problem Description

1.1. Problem Statement

In today's age of COVID, work from home has become omnipresent. A significant increase in the amount of Venture Capital money along with investor's increasing focus on asset-light business models has further enhanced this trend. In this environment, we plan to offer a timely solution that would help employers to remotely manage warehouses. Our software will help employees remotely monitor the inventory status, raise red flags and plan supplies, monitor costs, and work efficiently in teams. Our software will also offer safety features such as raising red flags whenever an external IP is encountered trying to access the system. It will aid managers to supervise employees remotely, monitor and ensure smooth operation in multiple teams, and control costs and glide towards an efficient inventory management system that avoids stock-outs at lowest cost possible.


1.2. Organization Description and goals

- Remote warehouse management

The warehouse management system is one of the most important components for the operations that manage product inventory or store manufactured goods and materials in physical stores. The warehouse management weaves a complex network of different areas, varied goods, and numerous clients. When the operating mode is working from home, the connections between staff and remote warehouses become challenging issues that require efficiency, low latency, and timeliness in the businesses. Our database solution provides access to the remote warehouses for people who work from home and helps to manage the remote warehouses efficiently.

We design our solution for client companies who are typically required to manage multiple warehouses to meet downstream demand. Each warehouse is identified with a unique ID, number of store sections, and capacity. Within one warehouse, the store sections have their own distinctive IDs and capacities. Besides, we offer an assembly function for one store section to combine with others if our client wants to store up the same inventory in different store sections. This function makes it possible to check all the locations of one specific inventory fast and organize the volumes deposited in different sections effectively.

Employees of our client company oversee the information of inventories at home for each trade by recording the information of goods or inventories, such as the number of items, suppliers, purchasers, date-in, date-out, costs, and selling prices, in the assembled sections. For instance, when our client purchases goods from a supplier, the employees should record the name of the supplier's company, purchasing date, the amount of goods acquired, trading prices, and other information. A trade table includes all the data created in this process plus the personal information, such as ID, department, phone number, and IP address of the employees who work on the data entry. When mistakes occur, it is convenient to review the trading information and find the supervisor based on the tables.



For security reasons, we establish tables recording the IT information, such as the IP addresses, for all the remote warehouses, company's server, and employees' home system, since the businesses are conducted online. For instance, a commonly used range of IP addresses for one employee are recorded in his/her personal information. When this employee is managing the warehouse online, we examine the current IP address first to avoid any potential risky operations. If the employee's IP address is not in the normal range, we will suspend his/her remote connection with the warehouse. In order to ensure a smooth operation of this remote work, our system will record information including the employee ID, name, beginning time, ending time, memory occupied, software used, and the IP address of each user.


Warehouse management could involve multiple employees working in a team. Consequently, as an end-to-end remote warehouse solution, it is imperative that our software help our clients to effectively work both within and between teams. To eliminate the inconvenience of working from home, we track records of the project progress by creating tables documenting the ID, name, description of the project, beginning time, finishing time, engaged teams or departments and project status. For example, under Just in time inventory management, procurement department of a firm may be directly coordinating with the design and manufacturing department. In this case, procurement department may be our key customer who could use our software to remotely manage inventory and our team-work feature may help all involved departments to manage work. Our software will also record information on teams or departments including entity name, member ID, member names, and number of staff. When bugs emerge during the test, our solution helps to quickly and precisely pinpoint the specific teams or employees that are responsible for the errors. For e.g. Process manufacturing critically depends upon successful conclusion of the previous process in order to commence. Our software will help the teams track progress of other teams and plan work.

- Ancillary services

Some of the companies or organizations have their own systems or digital working platforms. If a corporation subscribes for service of an expensive software installed in the system, the manager will allow the employees to run the software only in their office. Given, it is inconvenient for staff and teams who have demands for that software while work from home. Our solution provides our clients with a database solution of remote desktop usage. Workers run the software in the company's system, and our solution helps to keep track of the process to manage the load of the system.

When an employee or manager requires a remote usage, information including the employee ID, name, beginning time, ending time, memory occupied, software used, and the IP address of the user will be recorded in tables for future review. If too many people who work from home are running the remote desktop simultaneously, the capacity of the company's system may reach to its limit and that may raise the risk of suffering downtime. Based on our history records, it is straightforward for our clients to figure out the peak hour of remote operations when the burden on the company's system is too heavy and hence to adopt some measures that disperse the workers' operating hours.

We also help our clients to monitor the process of teamwork for team members who work from home. During the development of products or engagement of large projects, teams from different departments of the company



should coordinate with each other in the whole process. To eliminate the inconvenience of working from home, we track records of the project progress by creating tables documenting the ID, name, description of the project, beginning time, finishing time, engaged teams or departments, and details of failing cases if applicable. There are tables of information on teams or departments including entity name, member ID, member names, and number of staff as well. When bugs emerge during the test, our solution helps to quickly and precisely pinpoint the specific teams or employees that are responsible for the errors.

1.3. Business requirements

Following are the business requirements for goal 1: Data modeling

1. Design the Entity relation diagram of the business with the following key elements:
 - a. Base tables for Employee, warehouse, meeting, projects and remote login
 - b. Ability to match warehouse and section information tables with inventory log
 - c. Match employee login information to meeting records
 - d. Match employees and meetings with projects
2. Create the database with relevant entity tables and insert the current business data available
3. Generate tables to resolve following business requirements:
 - a. Ability to monitor capacity utilization at section/warehouse and promptly flag over/under-utilization
 - b. Remotely provide section codes to the client and change codes as section is leased to new project (client)
 - c. Track incoming and outgoing inventory
 - d. Track project progress
 - e. Remote team and project management :
 - Track employee remote login IP to flag inconsistencies and prevent cyber attacks
 - Coordinate virtual meeting attendance and concord with employee login register

1.4. Relational Data Model

Our model has 14 tables. Each table has its own primary key and is linked to one or more tables through foreign keys.

- Employee table has all information about firm employees including their name, address, gender etc.
 - In addition, it also covers an “IP Address range” which is allotted to each employee. This is used to trigger a flag if an employee logs in from a system outside of this IP range.
 - This table is linked to department table
- Department has information about each department including department manager
- Project information is stored in projects and project teams. The latter has projectID and employeeID as CPK

- Warehouse, and section tables have respective IDs and primary key and have information such as warehouse address and section capacity respectively
- Section codes table has section ID and project ID as primary key.
 - Thus as and when the section is rented by a new client (client of client!), the access codes are updated to preserve safety
- Log table records all incoming and outgoing inventories in a section under a project
 - Log table is linked to projects table via Project ID
- Meeting information is captured by meeting and meeting_attendance table
 - They also records project ID and participants for the meeting
- Remote login and client safety features, such as IP tracking, are captured by remoteDesk and remoteDeskWarning table

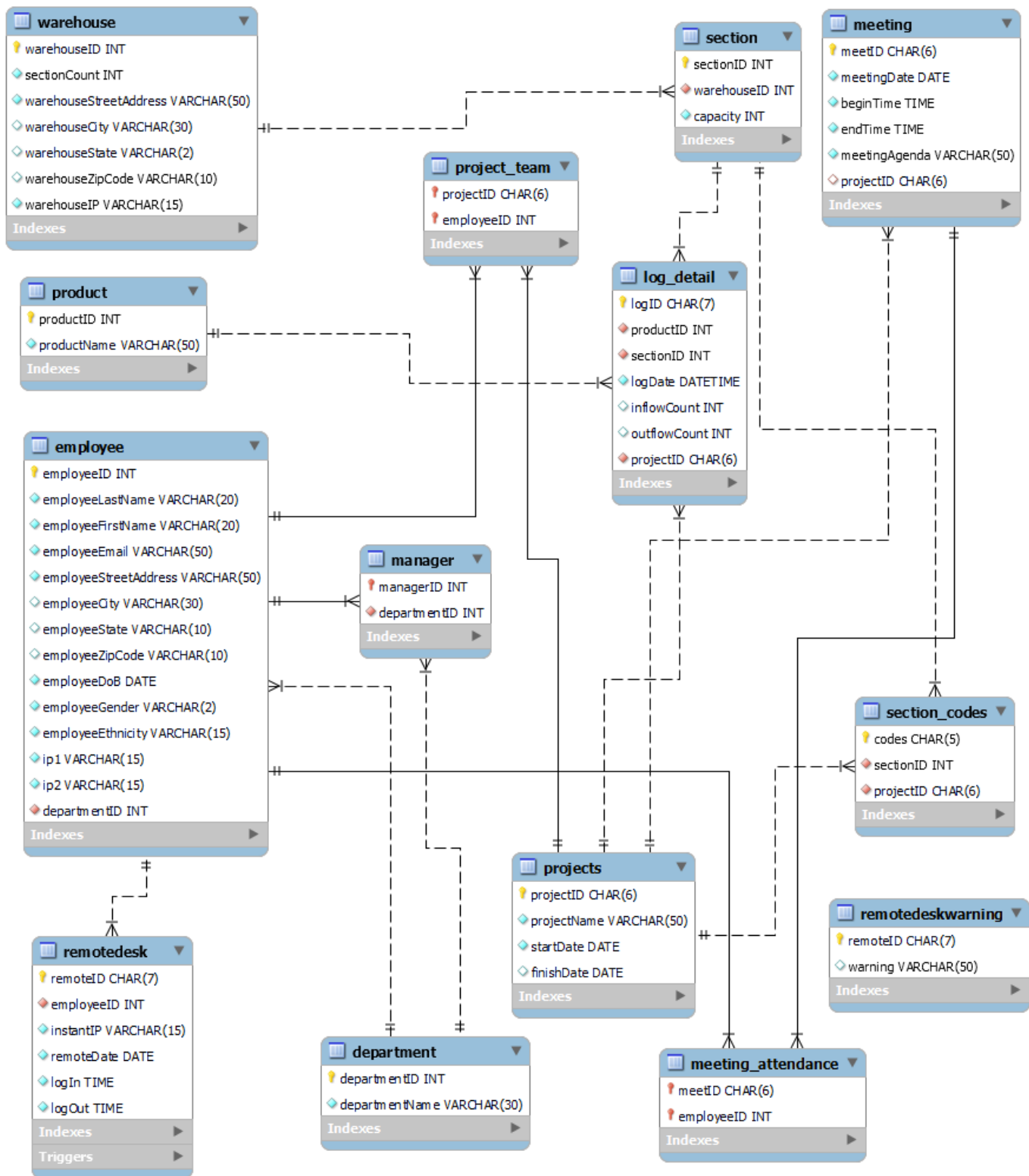
1.5. Assumptions about Data Entities and Relationships

- Each department has one manager.
 - Our queries are robust to a scenario with multiple department heads
- Employees from different department could participate in a project
- Meetings with “Internal meeting” agenda are admin meetings, for example hiring, office re-modelling etc
 - Employees from other departments can occasionally participate
- A warehouse can have multiple sections
 - A client can at a minimum rent the entire section
 - In other words, on any day, a section will carry inventory from a single client.
- Login after 12PM is counted half day
 - Logout after 6PM is counted overtime
- Section access codes are updated only once existing lease expires and a new client (project) rents the section

1.6. Scenarios covered for Database

- The database tracks warehouses managed remotely by our client
 - All incoming and outgoing inventory to each section are tracked and tagged to project ID
 - Clients (of our client) are sent section access code remotely and codes are updated whenever section is leased by new project
- It also allows client to manage work remotely
 - This includes tracking employee login, meetings, and project membership
- Our database embodies certain safety features as well such as:
 - Trigger activated whenever employees login from an unknown IP
- Project progress is covered via meeting_agenda which includes “meeting closing”, “opening”, “updates” etc.

2. Entity Relationship Diagram



3. Scope of the database

3.1.1 Employee Table

Table name	Primary key	Foreign key	Non-key attributes	# of Rows in Table
employee	employeeID	departmentID	employeeLastName employeeFirstName employeeEmail employeeStreetAddress employeeCity employeeState employeeZipCode employeeDoB employeeGender employeeEthnicity ip1 ip2	10

3.1.2 Department Table

Table name	Primary key	Foreign key	Non-key attributes	# of Rows in Table
department	departmentID		departmentName	3

3.1.3 Manager Table

Table name	Primary key	Foreign key	Non-key attributes	# of Rows in Table
manager	managerID	managerID departmentID		3

3.1.4 WarehouseTable

Table name	Primary key	Foreign key	Non-key attributes	# of Rows in Table
warehouse	warehouseID		sectionCount warehouseStreetAddress warehouseCity warehouseState warehouseZipCode warehouseIP	3

3.1.5 Section Table

Table name	Primary key	Foreign key	Non-key attributes	# of Rows in Table
section	sectionID	warehouseID	capacity	6

3.1.6 Product Table

Table name	Primary key	Foreign key	Non-key attributes	# of Rows in Table
product	productID		productName	8

3.1.7 Projects Table

Table name	Primary key	Foreign key	Non-key attributes	# of Rows in Table
projects	projectID		projectName startDate finishDate	5

3.1.8 Log detail Table

Table name	Primary key	Foreign key	Non-key attributes	# of Rows in Table
log_detail	logID	productID sectionID projectID	logDate inflowCount outflowCount	15

3.1.9 Remote Desk Table

Table name	Primary key	Foreign key	Non-key attributes	# of Rows in Table
remoteDesk	remoteID	employeeID	instantIP remoteDate logIn logOut	36

3.1.10 Remote Desk Warning Table

Table name	Primary key	Foreign key	Non-key attributes	# of Rows in Table
remoteDeskWarning	remoteID		warning	1

3.1.11 Meeting Table

Table name	Primary key	Foreign key	Non-key attributes	# of Rows in Table
meeting	meetID	projectID	meetingDate beginTime endTime meetingAgenda	12

3.1.12 Meeting Attendance Table

Table name	Primary key	Foreign key	Non-key attributes	# of Rows in Table
meeting_attendance	meetID employeeID	meetID employeeID		48

3.1.13 Project Team Table

Table name	Primary key	Foreign key	Non-key attributes	# of Rows in Table
project_team	projectID employeeID	projectID employeeID		21

3.1.14 Section Codes Table

Table name	Primary key	Foreign key	Non-key attributes	# of Rows in Table
section_codes	codes	sectionID projectID		8

3.2 3NF DATABASE

Our database is 3NF since we've taken due care that:

- All tables are 1NF: All cells contain only scalar values
- All tables are 2NF: 1NF and all non-key attributes in a table are functionally dependent upon entire primary key
 - For e.g. departmentID is a non-key attribute in employee table which is functionally dependent entirely on employeeID (primary key)
 - Similarly "sectioncount" is non-key attribute in warehouse table which is fully functionally dependent upon warehouseID
- All table are 3NF: 2NF and there is no transitive dependency on primary key
 - For e.g. even though employee ethnicity is a non-key attribute, it is directly (and NOT transitively through some other non-key attribute), dependent upon employeeID (Primary key)
 - Similarly "outflowCount" and "inflowCount" in the log_detail table are directly dependent upon logID and not transitively.

4. Retrieving the data according to business goals

4.1. Query 1

When Walmart delivered some items to the warehouse located in Chestnut Hill, which sections were used, what security codes were sent to the delivery staff, and how many times has the section been accessed?

Query

```
select distinct s.sectionID, codes, count(logID) as accessTimes from log_detail ld
inner join section s on s.sectionID = ld.sectionID
inner join warehouse w on w.warehouseID = s.warehouseID
inner join projects p on ld.projectID = p.projectID
inner join section_codes sc on sc.projectID = ld.projectID and sc.sectionID = ld.sectionID
where p.projectName like 'Walmart%'
and w.warehouseCity = 'Chestnut Hill';
```

Output:

	sectionID	codes	accessTimes
▶	5	347q8	4

4.2. Query 2 (Stored Function)

What is the capacity utilization for the warehouse at NY on 7 July 2022?

Query

```
DROP FUNCTION IF EXISTS get_warehouse_cap;
DELIMITER $$
create function get_warehouse_cap(
warehouse_state varchar(10)
)
returns int
deterministic
begin
declare warehouse_cap int;
select sum(capacity)
into warehouse_cap
from section
inner join warehouse on warehouse.warehouseID = section.warehouseID
where warehouse.warehouseState = warehouse_state ;
return warehouse_cap;
end$$
delimiter ;

USE wfh2db;
```

```

SELECT w.warehouseState, get_warehouse_cap('NY') as total_cap, sum(ld.inflowCount-ld.outflowCount) as
utilization, sum(ld.inflowCount-ld.outflowCount) / get_warehouse_cap('NY') as cap_utilization
FROM warehouse w
JOIN section s ON s.warehouseID = w.warehouseID
JOIN log_detail ld ON ld.sectionID = s.sectionID
WHERE w.warehouseState = 'NY'
AND ld.logDate <= '2022-07-07'
group by w.warehouseState;

```

Output:

	warehouseState	total_cap	utilization	cap_utilization
▶	NY	130	55	0.4231

4.3. Query 3

How many total iphone 13 were present in Amazon inventory as on 7 July 2022?

Query

```

SELECT pd.productName,sum(ld.inflowCount-ld.outflowCount) as tot_count FROM log_detail ld
JOIN projects p ON ld.projectID = p.projectID
JOIN product pd ON pd.productID = ld.productID
WHERE p.projectName LIKE 'Amazon%'
AND pd.productName = 'iphone13'
AND ld.logDate <= '2022-07-07'
GROUP BY ld.productID;

```

Output:

Result Grid	Filter Rows:
productName	tot_count
▶ iphone13	55

Explanation:

Our client maintains strict confidentiality while recording customer's inventory data (sections). For example: Amazon's inventory related data is accessible **only** by Amazon. Customers like Amazon frequently request our client to provide them updated records of their inventory.

4.4. Query 4 (Views)

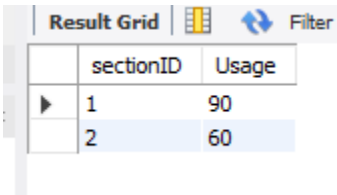
Show the latest usage of sections located in Queens, NY.

Query

```
DROP VIEW IF EXISTS section_usage_view;
CREATE VIEW section_usage_view AS
SELECT ld.sectionID,(s.capacity+sum(ld.inflowCount)-sum(ld.outflowCount)) AS 'Usage' FROM section s
JOIN log_detail ld ON s.sectionID = ld.sectionID
GROUP BY ld.sectionID;

SELECT * FROM section_usage_view
where sectionID in
(select section.sectionID from
section join warehouse
on section.warehouseID = warehouse.warehouseID
where warehouseCity = 'Queens' and
warehouseState = 'NY');
```

Output:



	sectionID	Usage
▶	1	90
	2	60

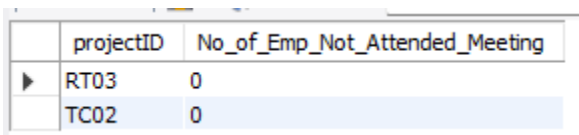
4.5. Query 5

How many employees, who were part of the project team, were absent from the "project closing" meeting held between 7 July-8 July?

Query

```
SELECT m.projectID,count(e.employeeID) NOT IN
(SELECT count(employeeID) FROM project_team
WHERE projectID IN (SELECT projectID FROM meeting
WHERE meetingAgenda LIKE '%project closing')
GROUP BY projectID) as No_of_Emp_Not_Attended_Meeting FROM employee e
JOIN project_team pt ON pt.employeeID = e.employeeID
JOIN meeting m ON m.projectID = pt.projectID
WHERE m.meetingAgenda LIKE '%project closing'
AND m.meetingDate BETWEEN '2022-07-07' AND '2022-07-08'
GROUP BY m.meetID;
```

Output:



	projectID	No_of_Emp_Not_Attended_Meeting
▶	RT03	0
	TC02	0

4.6. Query 6

Between 4 Jul 2022 - 8 Jul 2022, how many internal meetings were held with non-administrative employees attending ?

Query

```
SELECT m.meetingDate, count(m.meetingAgenda) as count
FROM meeting as m INNER JOIN meeting_attendance as ma
ON m.meetID = ma.meetID
INNER JOIN employee as e
ON ma.employeeID = e.employeeID
INNER JOIN department as d ON e.departmentID = d.departmentID
WHERE m.meetingAgenda = 'Internal meet'
AND d.departmentName NOT LIKE 'Administration%'
AND meetingDate BETWEEN '2022-07-04' AND '2022-07-08'
GROUP BY m.meetingDate;
```

Output:

	meetingDate	count
▶	2022-07-05	1

4.7. Query 7

On 7 July 2022 which employees did not log in even though they had a project meeting that day?

Query

```
SELECT distinct ma.employeeID FROM meeting_attendance ma
JOIN meeting m ON m.meetID = ma.meetID
where m.meetingDate = '2022-07-07'
and ma.employeeID Not IN (SELECT r.employeeID FROM remotedesk r
WHERE r.remoteDate = '2022-07-07');
```

Output:

	employeeID
▶	1010

4.8. Query 8

Between 5 July 2022 - 8 July 2022 how many employees logged in half day (after 12PM) in order to attend a meeting?

Query


```
SELECT count(distinct rd.employeeID) AS emp_count FROM remotedesk rd
join meeting_attendance ma on ma.employeeID = rd.employeeID
join meeting m on m.meetID = ma.meetID and m.meetingDate = rd.remoteDate
WHERE remoteDate BETWEEN '2022-07-05' AND '2022-07-08'
AND logIn > '12:00:00';
```

Output:

emp_count
4

4.9. Query 9 (Trigger)

Are there any abnormal logins? If so, list the employee ID, the login time, and the abnormal IP address.

Query

```
DROP TRIGGER IF EXISTS ip_warning;
DELIMITER $$
CREATE TRIGGER ip_warning after insert on remoteDesk
FOR EACH ROW
BEGIN
IF (strcmp(trim(trailing SUBSTRING_INDEX(NEW.instantIP, '.', -1) from NEW.instantIP),
trim(trailing SUBSTRING_INDEX((select ip1 from employee where new.employeeID = employeeID), '.', -1) from
(select ip1 from employee where new.employeeID = employeeID))) != 0)
THEN
insert into remoteDeskWarning (remoteID, warning)
values (new.remoteID, 1);
END IF;
END$$
DELIMITER ;

select employeeID, remoteDate, logIn, logOut, instantIP from remoteDesk as rd
right join remoteDeskWarning as rdw
on rd.remoteID = rdw.remoteID
```

Output:

employeeID	remoteDate	logIn	logOut	instantIP
1001	2022-07-07	08:05:40	16:50:42	103.54.152.245

4.10. Query 10 (Stored Function & Stored Procedure)

As a small firm looking to raise funds, our client cares about the social impact. Hence they want to know how many days on average did the female department heads logged in versus male department heads between 6 July

2022 - 7 July 2022?

Query

```
DROP FUNCTION IF EXISTS get_dept_head;
DELIMITER $$
CREATE FUNCTION get_dept_head(
emp_ID int,
gender char
)
RETURNS int
DETERMINISTIC
BEGIN
DECLARE emp_Count int;
SELECT count(employeeID) INTO emp_Count
FROM manager m
JOIN employee e ON e.employeeID = m.managerID
WHERE e.employeeGender = gender
GROUP BY e.employeeGender;
RETURN emp_Count;
END$$
DELIMITER ;

DROP PROCEDURE IF EXISTS dept_head_log;
DELIMITER //
CREATE PROCEDURE dept_head_log()
BEGIN
select distinct (count(employeeID)/get_dept_head(employeeID,'F')) as avg_wo_days,
(count(employeeID)/get_dept_head(employeeID,'M')) as avg_man_days
FROM remotedesk
WHERE employeeID IN (SELECT managerID FROM manager)
AND remotedate BETWEEN '2022-07-06' AND '2022-07-07'
GROUP BY employeeID;
END //
DELIMITER ;

CALL dept_head_log();
```

Output:

Result Grid		Filter Rows:
avg_wo_days	avg_man_days	
1.0000	2.0000	

4.11 Query 11

List non-white employees who worked overtime (post 6PM) between 4 Jul 2022 - 8 July 2022.

Query

```
SELECT e.employeeFirstName, e.employeeLastName, e.employeeEthnicity, rd.remoteDate, rd.logOut
FROM employee as e INNER JOIN remotedesk as rd
ON e.employeeID = rd.employeeID
WHERE (remoteDate BETWEEN '2022-07-04' AND '2022-07-08') AND (logOut > '18:00:00') AND
(employeeEthnicity != 'White');
```

Output:

Result Grid					
Filter Rows: <input type="text"/>					
Export: <input type="button" value="Export"/>					
Wrap Cell Content: <input type="button" value="Wrap"/>					
	employeeFirstName	employeeLastName	employeeEthnicity	remoteDate	logOut
▶	Tiffany	Austin	Black	2022-07-05	18:50:59
	Nancy	Young	Asian	2022-07-06	18:50:59
	Michael	Clark	Black	2022-07-07	19:00:36
	Jerry	Pinto	Hispanic	2022-07-07	19:01:00
	Nancy	Young	Asian	2022-07-07	18:50:42
	Jerry	Pinto	Hispanic	2022-07-08	18:00:55
	Nancy	Young	Asian	2022-07-08	19:00:36
	Tiffany	Austin	Black	2022-07-08	18:32:36

Explanation: SEC (Securities and Exchange Commission) recently mandated companies to disclose their human capital which includes diversity. Hence, our client wants to monitor its diversity.

- End of Document -