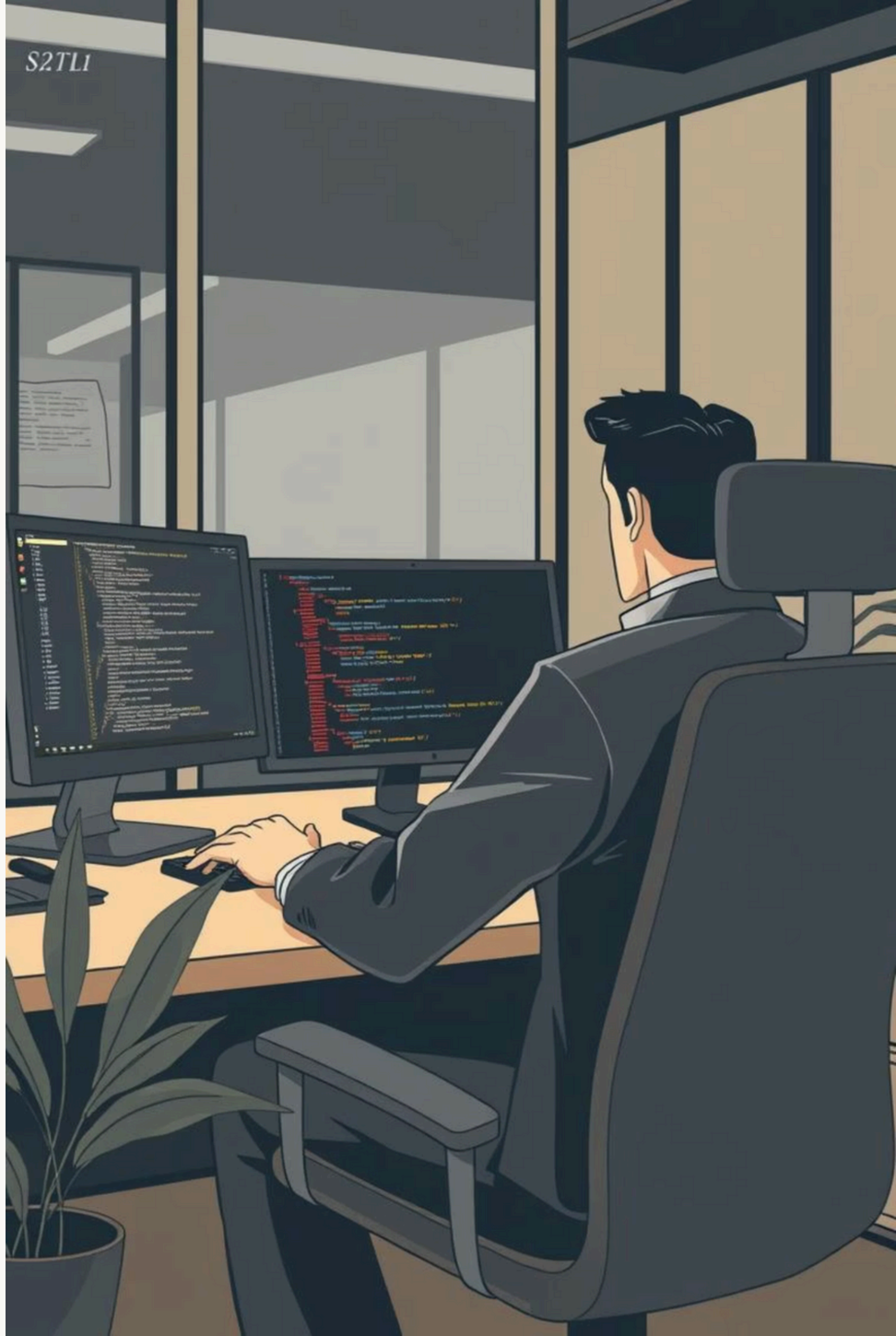
SQL CONCEPTS FOR DATA ANALYSTS & DATA SCIENCE

Unlock the power of data with SQL. This presentation covers fundamental concepts essential for every aspiring data analyst.

Name: Lucky Dubey

IMARTICUS LEARNING



THE FOUNDATION: SELECT & WHERE

SELECT: Choose Your Data

The **SELECT** statement is used to retrieve data from a database. It's the first step in querying information.

```
SELECT column1, column2 FROM table_name;
```

WHERE: Filter Your Results

The **WHERE** clause filters records based on specified conditions, allowing you to narrow down your results.

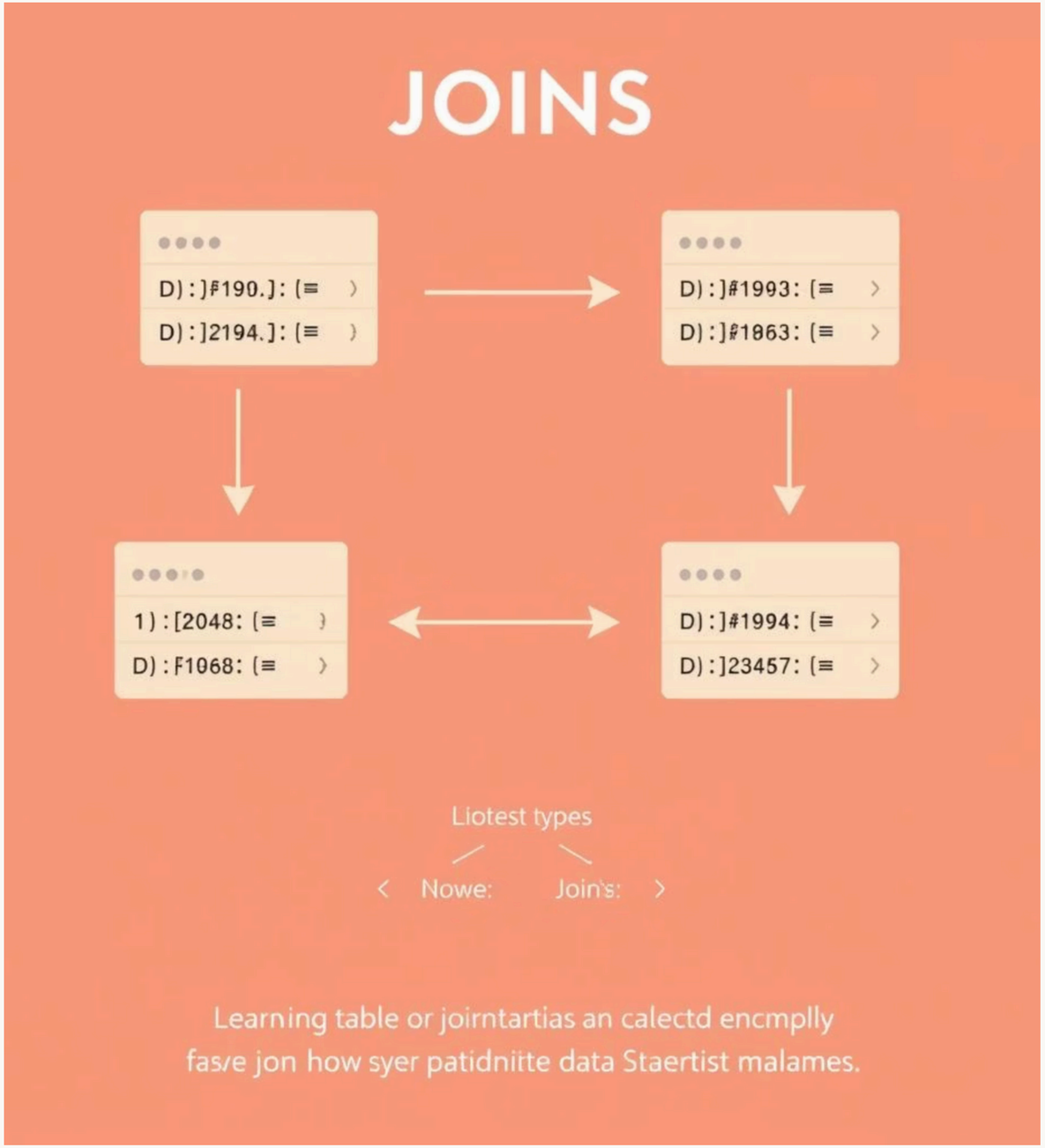
```
SELECT * FROM customers WHERE city = 'New York';
```

COMBINING DATA: JOINS

JOINS are crucial for combining rows from two or more tables based on a related column between them.

Types of JOINS

- **INNER JOIN**: Returns records that have matching values in both tables.
- **LEFT JOIN** (or LEFT OUTER JOIN): Returns all records from the left table, and the matched records from the right table.
- **RIGHT JOIN** (or RIGHT OUTER JOIN): Returns all records from the right table, and the matched records from the left table.
- **FULL JOIN** (or FULL OUTER JOIN): Returns all records when there is a match in either left or right table.



ORGANIZING AND SUMMARIZING: GROUP BY & ORDER BY

GROUP BY: Aggregate Your Data

The **GROUP BY** clause groups rows that have the same values into summary rows, often used with aggregate functions like COUNT, MAX, MIN, SUM, AVG.

```
SELECT country, COUNT(customer_id)FROM
customersGROUP BY country;
```

ORDER BY: Sort Your Results

The **ORDER BY** keyword is used to sort the result-set in ascending (**ASC**) or descending (**DESC**) order.

```
SELECT product_name, priceFROM
productsORDER BY price DESC;
```

DATA MANIPULATION: INSERT, UPDATE, DELETE

These commands allow you to modify the data within your database tables.

INSERT: Add New Data

Used to add new rows of data into a table. You must specify the table and the values for each column.

```
INSERT INTO customers
(customer_name, city)VALUES
('Alice Smith', 'London');
```

UPDATE: Modify Existing Data

Used to change existing data in one or more rows of a table. Always use a WHERE clause to avoid updating all rows.

```
UPDATE productsSET price =
25.00WHERE product_id = 101;
```

DELETE: Remove Data

Used to remove one or more rows from a table. Be cautious; DELETE permanently removes data.

```
DELETE FROM employeesWHERE
employee_id = 205;
```

ENSURING DATA INTEGRITY: KEYS & INDEXES

PRIMARY KEY

A unique identifier for each record in a table, ensuring no duplicate rows and quick data retrieval.

FOREIGN KEY

A field in one table that refers to the PRIMARY KEY in another table, establishing relationships.

INDEXES

Special lookup tables that the database search engine can use to speed up data retrieval.

ADVANCED TECHNIQUES: VIEWS & SUBQUERIES

VIEWS: Virtual Tables

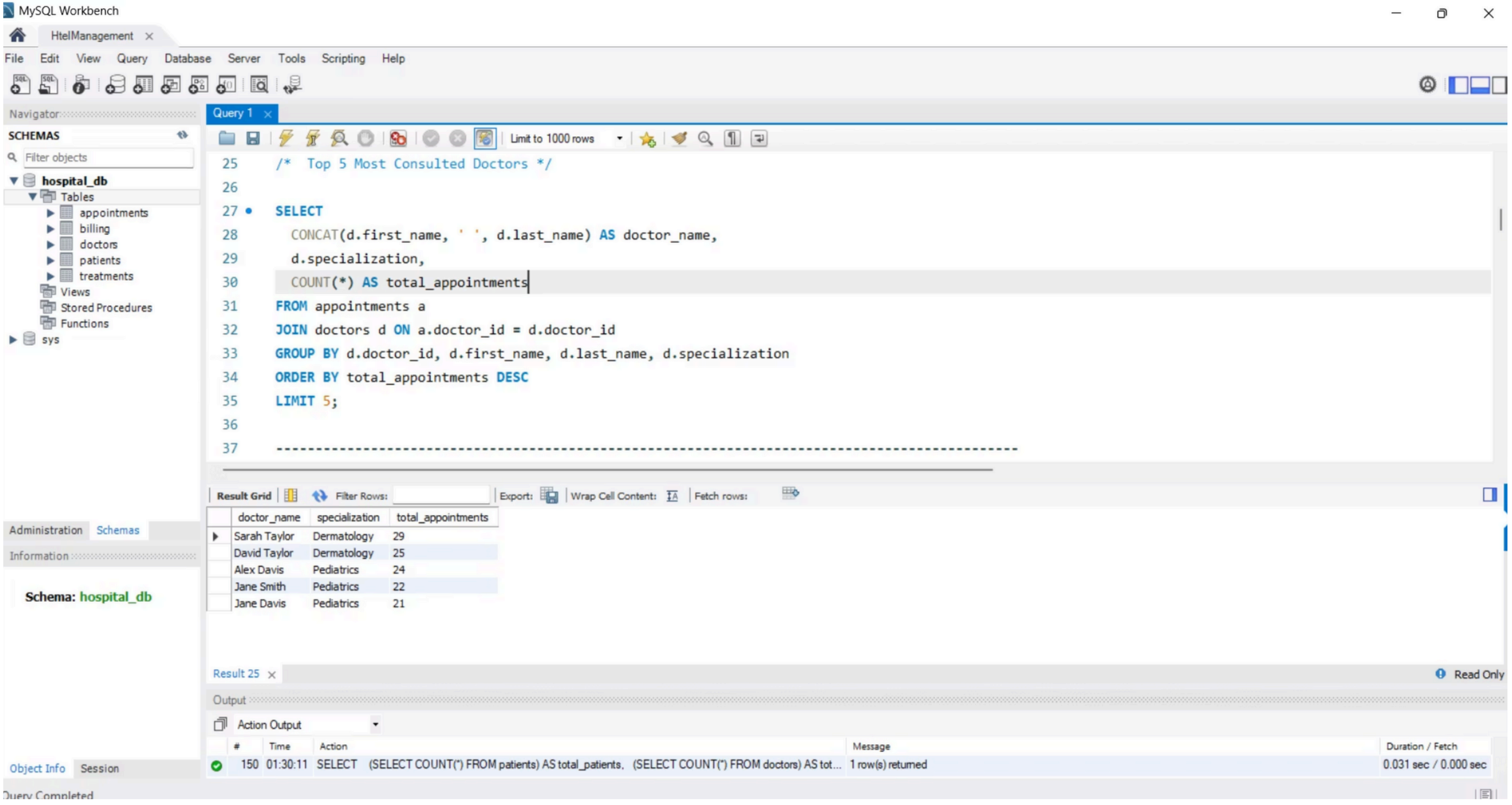
A **VIEW** is a virtual table based on the result-set of an SQL query. It contains rows and columns, just like a real table, but its data is not physically stored.

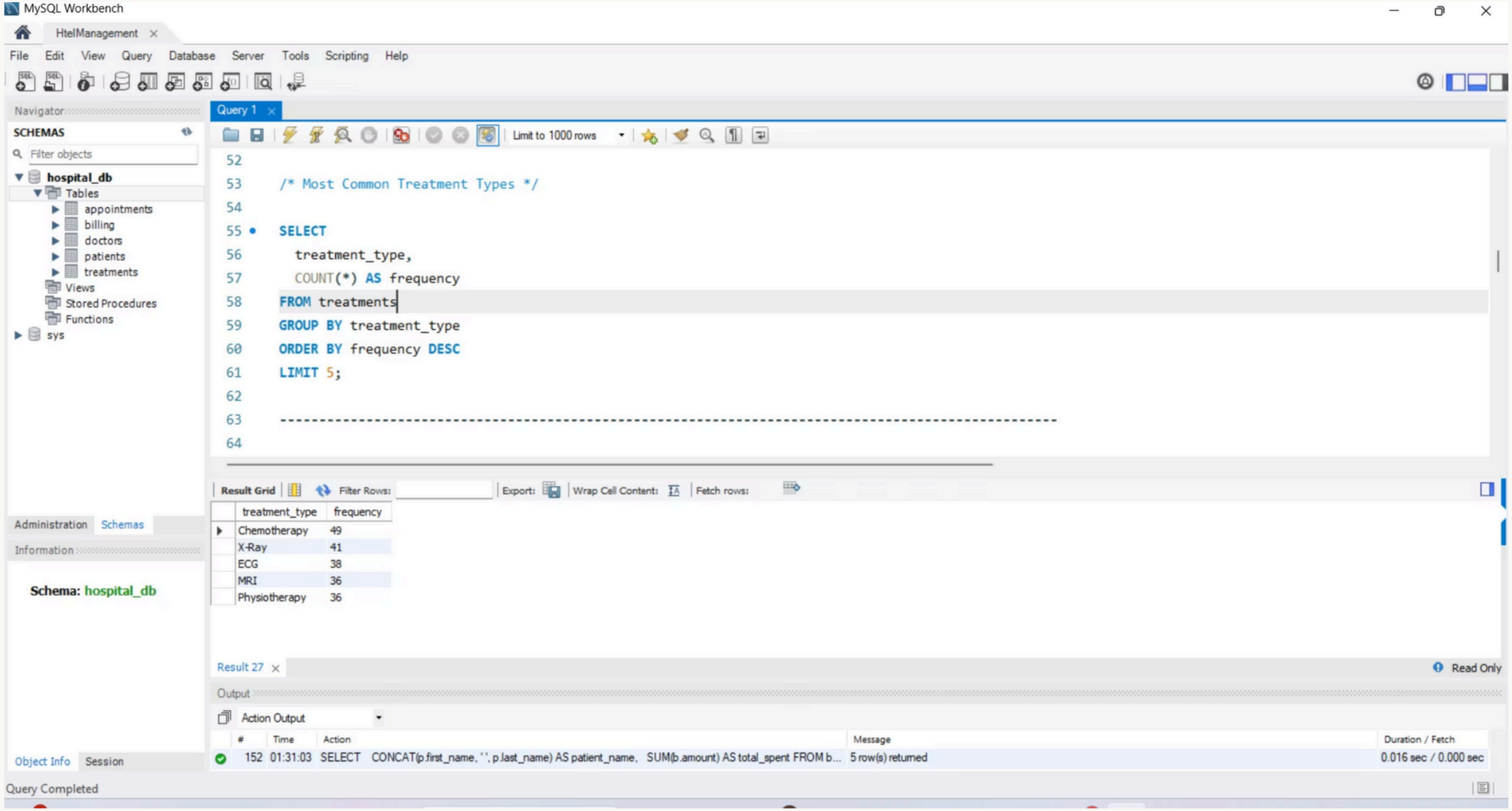
```
CREATE VIEW customer_orders ASSELECT
c.customer_name, o.order_idFROM customers c
JOIN orders o ON c.customer_id =
o.customer_id;
```

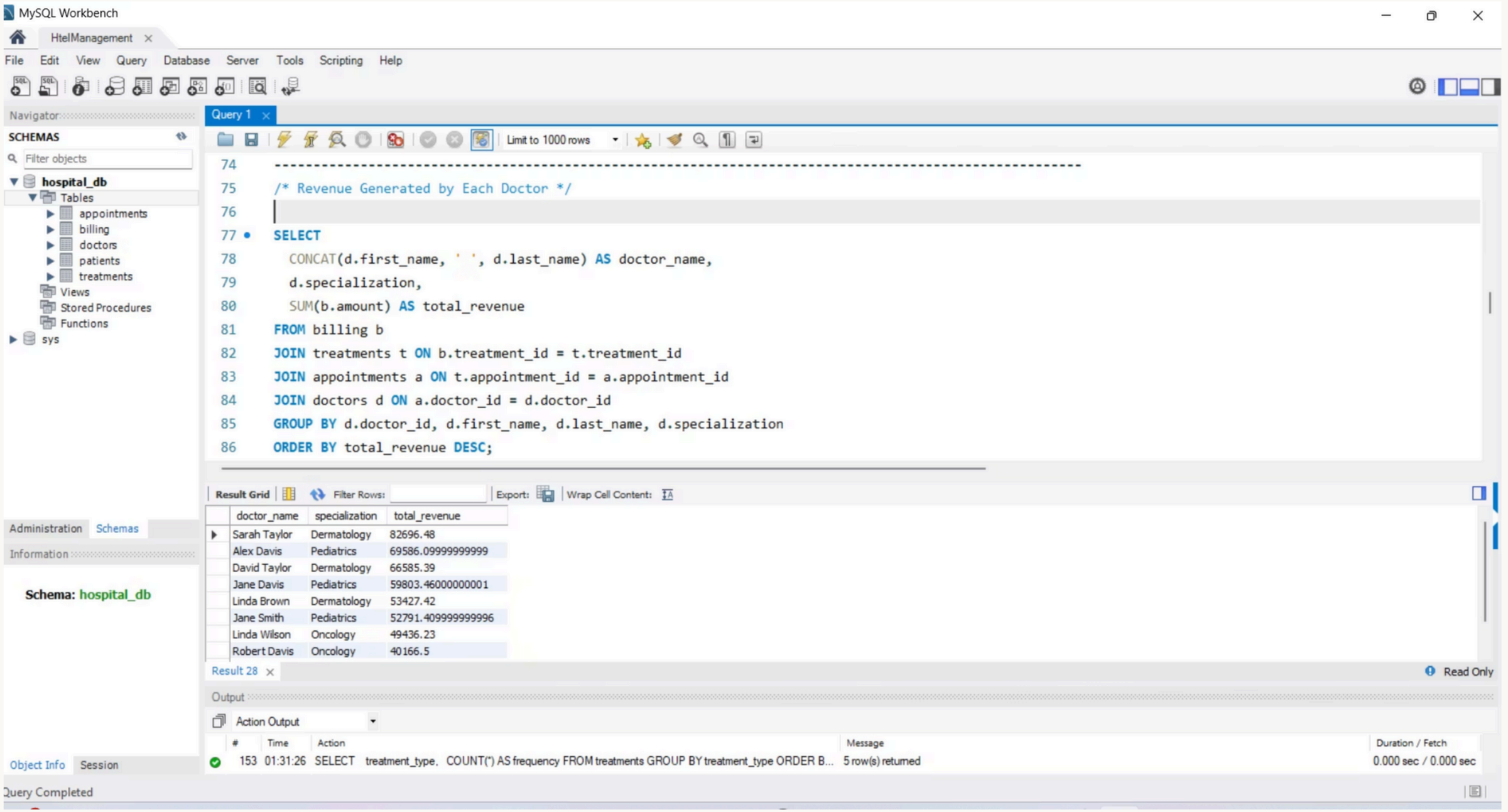
SUBQUERIES: Nested Queries

A **SUBQUERY** (or inner query) is a query nested inside another SQL query. It is executed first, and its result is used by the outer query.

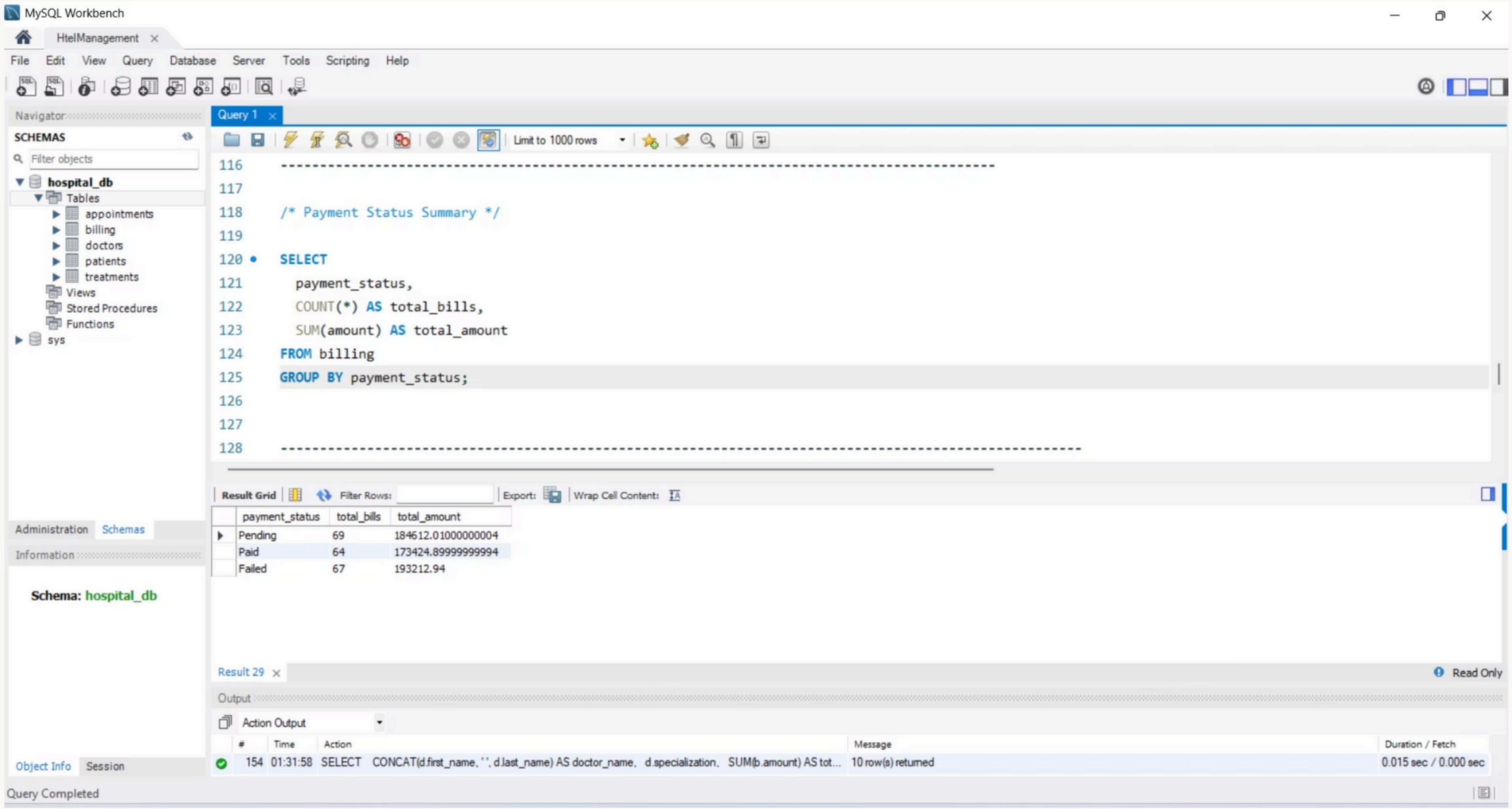
```
SELECT employee_nameFROM employeesWHERE
salary > (SELECT AVG(salary) FROM employees);
```

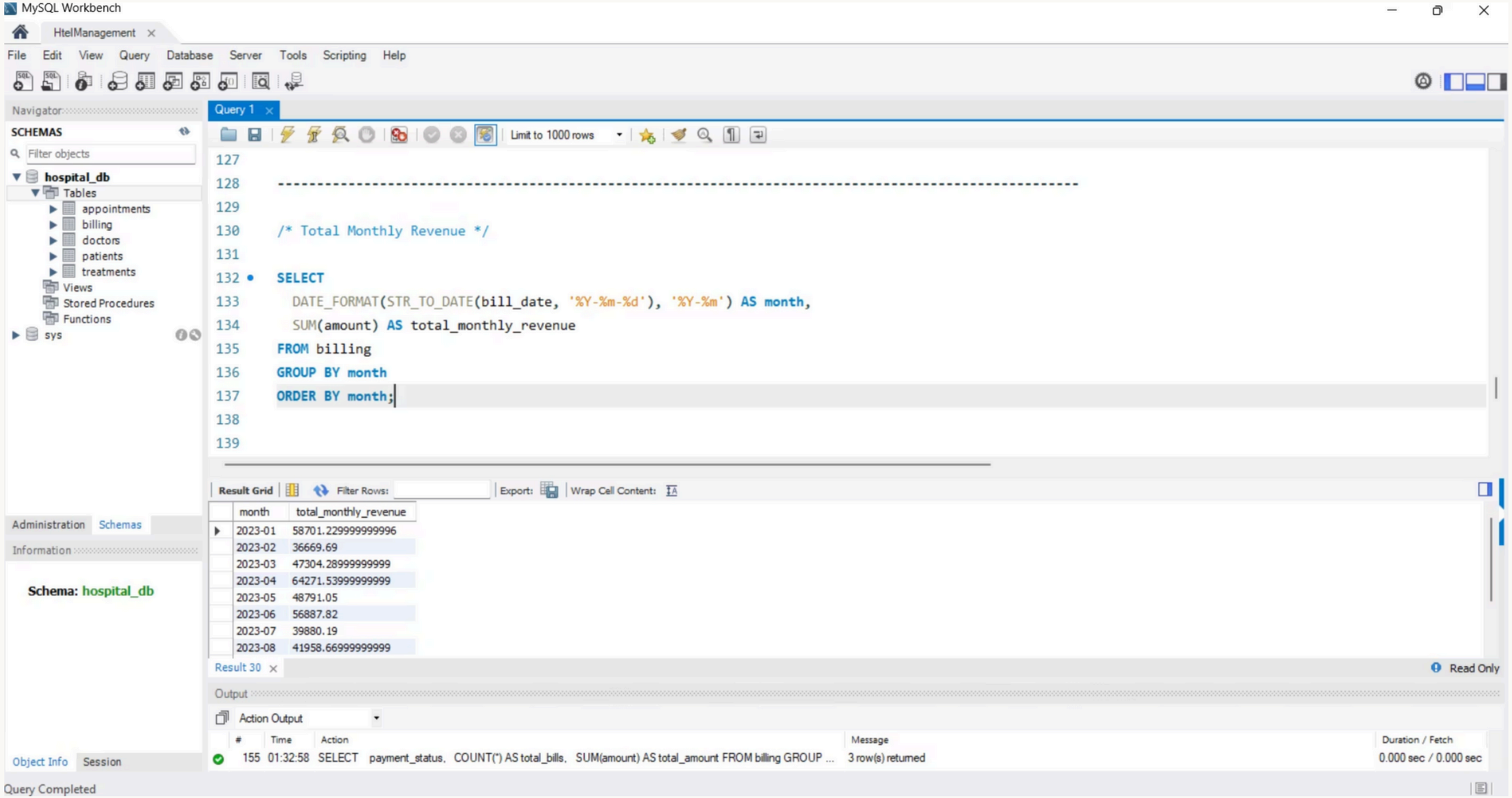

MySQL Workbench
HtelManagement
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
hospital_db
Tables
appointments
billing
doctors
patients
treatments
Views
Stored Procedures
Functions
sys
Administration Schemas
Information
Schema: hospital_db
Object Info Session
Query 1
Limit to 1000 rows
25 /* Top 5 Most Consulted Doctors */
26
27 SELECT
28 CONCAT(d.first_name, ' ', d.last_name) AS doctor_name,
29 d.specialization,
30 COUNT(*) AS total_appointments
31 FROM appointments a
32 JOIN doctors d ON a.doctor_id = d.doctor_id
33 GROUP BY d.doctor_id, d.first_name, d.last_name, d.specialization
34 ORDER BY total_appointments DESC
35 LIMIT 5;
36
37 ----------
Result Grid Filter Rows: Export: Wrap Cell Content: Fetch rows:
doctor_name specialization total_appointments
Sarah Taylor Dermatology 29
David Taylor Dermatology 25
Alex Davis Pediatrics 24
Jane Smith Pediatrics 22
Jane Davis Pediatrics 21
Result 25
Read Only
Output
Action Output
Time Action Message Duration / Fetch
150 01:30:11 SELECT (SELECT COUNT(*) FROM patients) AS total_patients, (SELECT COUNT(*) FROM doctors) AS tot... 1 row(s) returned 0.031 sec / 0.000 sec
Query Completed


MySQL Workbench
HtelManagement
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
hospital_db
Tables
appointments
billing
doctors
patients
treatments
Views
Stored Procedures
Functions
sys
Administration Schemas
Information
Schema: hospital_db
Object Info Session
Query 1
Limit to 1000 rows
52
53 /* Most Common Treatment Types */
54
55 SELECT
56 treatment_type,
57 COUNT(*) AS frequency
58 FROM treatments
59 GROUP BY treatment_type
60 ORDER BY frequency DESC
61 LIMIT 5;
62
63 ----------
64
Result Grid Filter Rows: Export: Wrap Cell Content: Fetch rows:
treatment_type frequency
Chemotherapy 49
X-Ray 41
ECG 38
MRI 36
Physiotherapy 36
Result 27
Read Only
Output
Action Output
Time Action Message Duration / Fetch
152 01:31:03 SELECT CONCAT(p.first_name, ' ', p.last_name) AS patient_name, SUM(b.amount) AS total_spent FROM b... 5 row(s) returned 0.016 sec / 0.000 sec
Query Completed

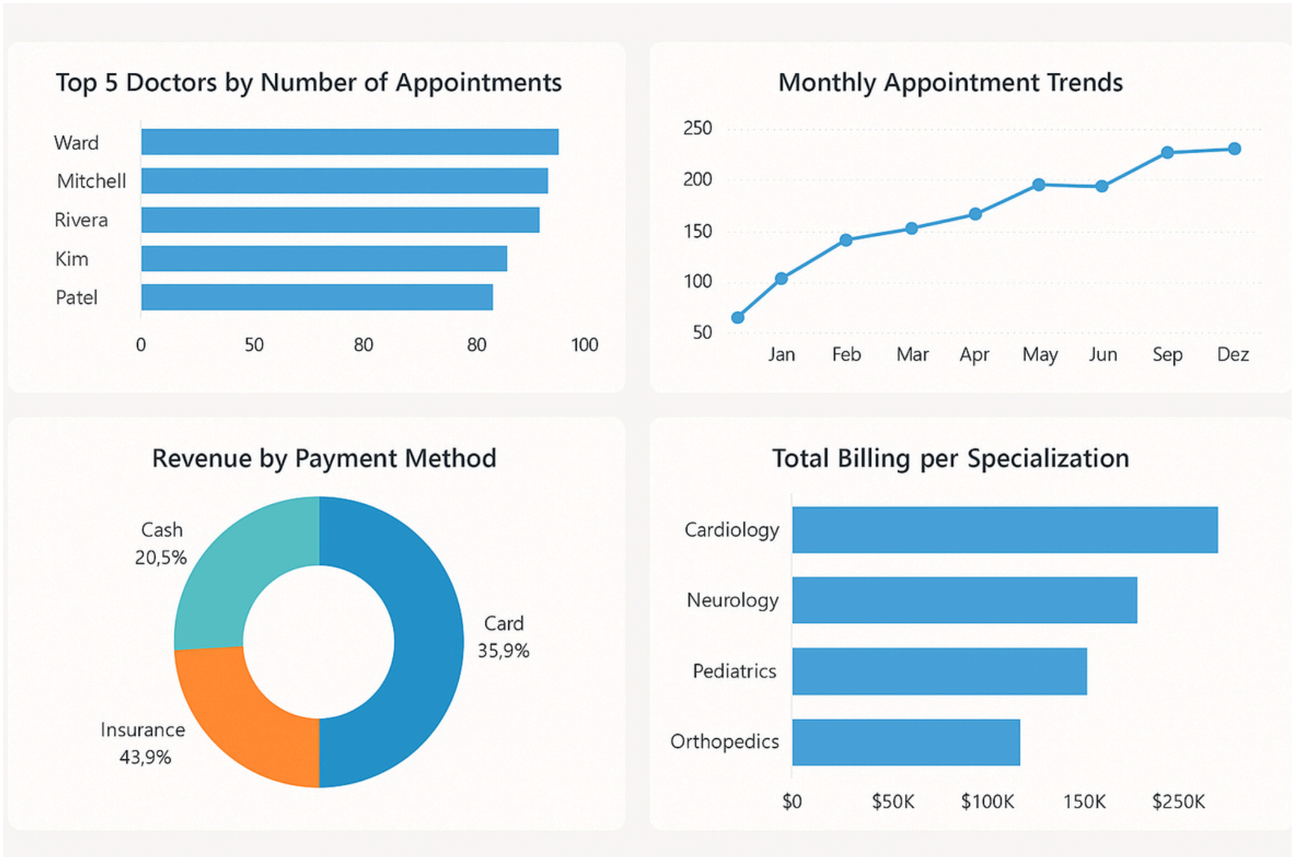

MySQL Workbench
HtelManagement
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
hospital_db
Tables
appointments
billing
doctors
patients
treatments
Views
Stored Procedures
Functions
sys
Administration Schemas
Information
Schema: hospital_db
Object Info Session
Query 1
Limit to 1000 rows
74 ----------
75 /* Revenue Generated by Each Doctor */
76
77 SELECT
78 CONCAT(d.first_name, ' ', d.last_name) AS doctor_name,
79 d.specialization,
80 SUM(b.amount) AS total_revenue
81 FROM billing b
82 JOIN treatments t ON b.treatment_id = t.treatment_id
83 JOIN appointments a ON t.appointment_id = a.appointment_id
84 JOIN doctors d ON a.doctor_id = d.doctor_id
85 GROUP BY d.doctor_id, d.first_name, d.last_name, d.specialization
86 ORDER BY total_revenue DESC;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
doctor_name specialization total_revenue
Sarah Taylor Dermatology 82696.48
Alex Davis Pediatrics 69586.09999999999
David Taylor Dermatology 66585.39
Jane Davis Pediatrics 59803.46000000001
Linda Brown Dermatology 53427.42
Jane Smith Pediatrics 52791.409999999996
Linda Wilson Oncology 49436.23
Robert Davis Oncology 40166.5
Result 28
Read Only
Output
Action Output
Time Action Message Duration / Fetch
153 01:31:26 SELECT treatment_type, COUNT(*) AS frequency FROM treatments GROUP BY treatment_type ORDER B... 5 row(s) returned 0.000 sec / 0.000 sec
Query Completed

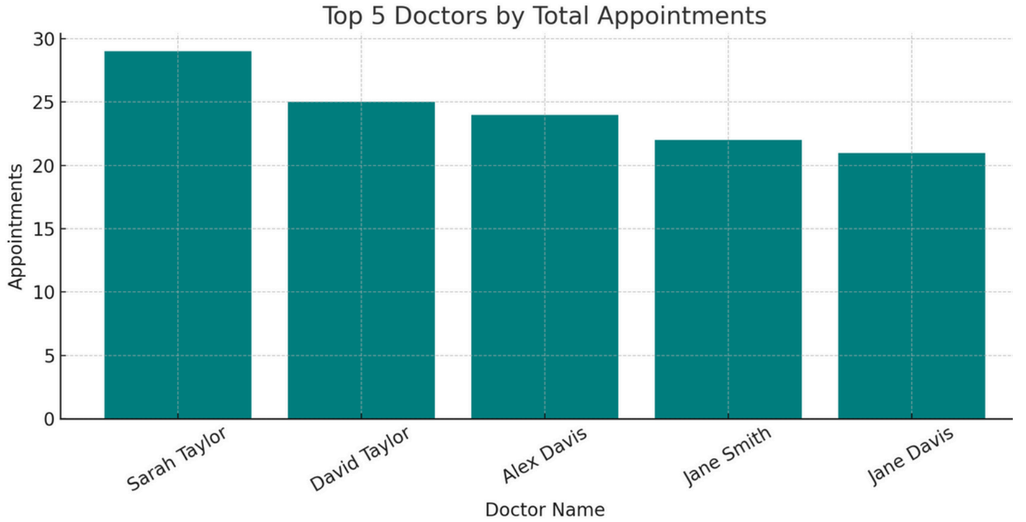

MySQL Workbench
HtelManagement

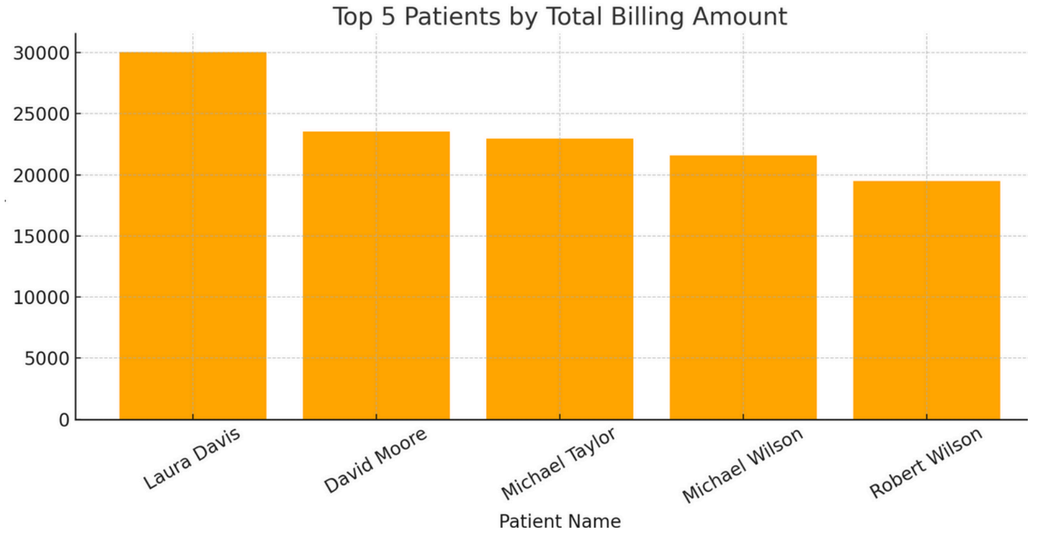

MySQL Workbench
HtelManagement
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
hospital_db
Tables
appointments
billing
doctors
patients
treatments
Views
Stored Procedures
Functions
sys
Administration Schemas
Information
Schema: hospital_db
Object Info Session
Query 1
Limit to 1000 rows
116 ----------
117
118 /* Payment Status Summary */
119
120 SELECT
121 payment_status,
122 COUNT(*) AS total_bills,
123 SUM(amount) AS total_amount
124 FROM billing
125 GROUP BY payment_status;
126
127
128 ----------
Result Grid Filter Rows: Export: Wrap Cell Content:
payment_status total_bills total_amount
Pending 69 184612.01000000004
Paid 64 173424.89999999994
Failed 67 193212.94
Result 29
Read Only
Output
Action Output
Time Action Message Duration / Fetch
154 01:31:58 SELECT CONCAT(d.first_name, ' ', d.last_name) AS doctor_name, d.specialization, SUM(b.amount) AS tot... 10 row(s) returned 0.015 sec / 0.000 sec
Query Completed


MySQL Workbench
HtelManagement
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
hospital_db
Tables
appointments
billing
doctors
patients
treatments
Views
Stored Procedures
Functions
sys
Administration Schemas
Information
Schema: hospital_db
Object Info Session
Query 1
Limit to 1000 rows
127
128 ----------
129
130 /* Total Monthly Revenue */
131
132 SELECT
133 DATE_FORMAT(STR_TO_DATE(bill_date, '%Y-%m-%d'), '%Y-%m') AS month,
134 SUM(amount) AS total_monthly_revenue
135 FROM billing
136 GROUP BY month
137 ORDER BY month;
138
139
Result Grid Filter Rows: Export: Wrap Cell Content:
month total_monthly_revenue
2023-01 58701.229999999996
2023-02 36669.69
2023-03 47304.28999999999
2023-04 64271.53999999999
2023-05 48791.05
2023-06 56887.82
2023-07 39880.19
2023-08 41958.66999999999
Result 30
Read Only
Output
Action Output
Time Action Message Duration / Fetch
155 01:32:58 SELECT payment_status, COUNT(*) AS total_bills, SUM(amount) AS total_amount FROM billing GROUP ... 3 row(s) returned 0.000 sec / 0.000 sec
Query Completed

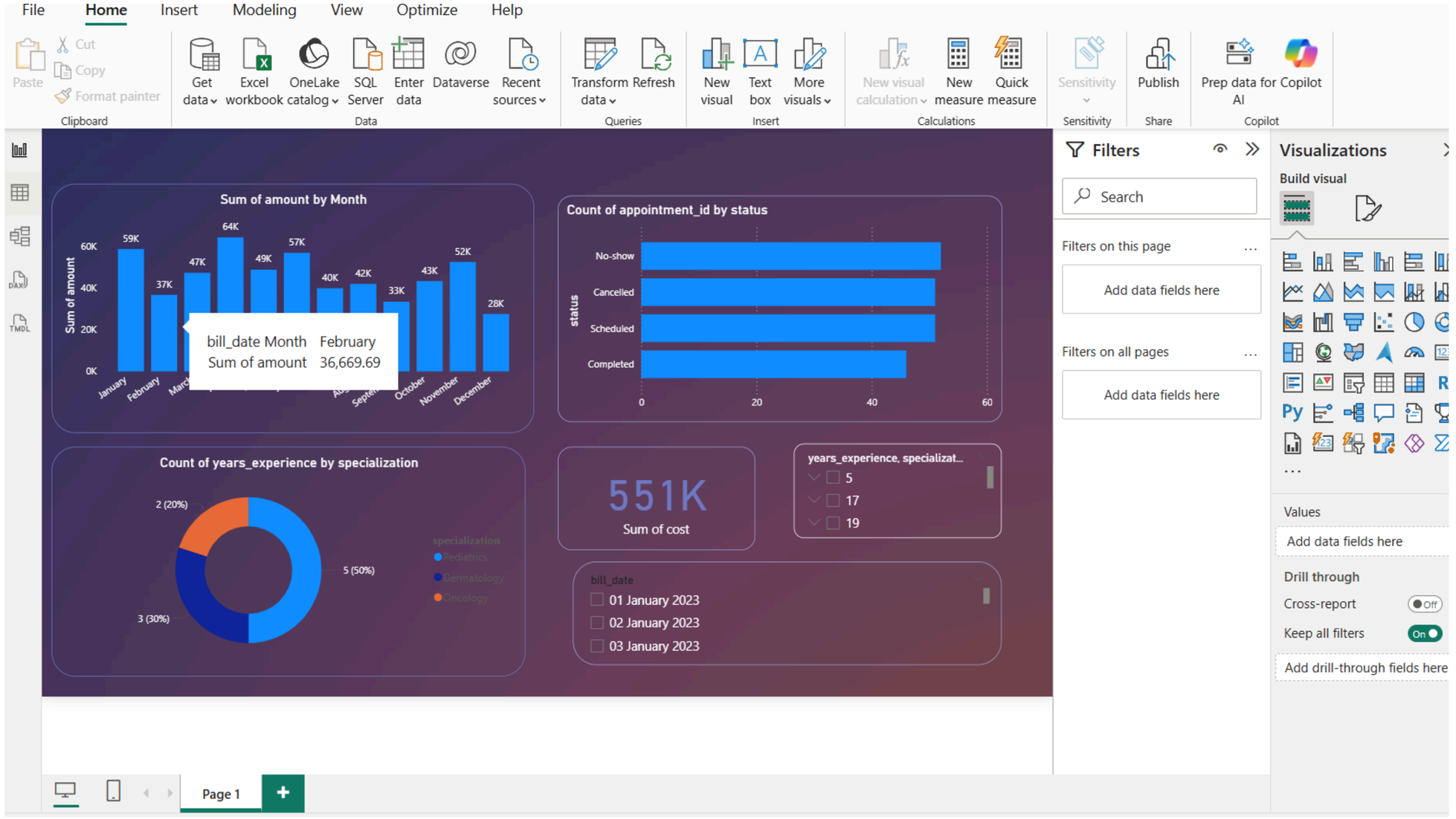
DATA ANALYSIS BY GRAPH & CHART VISUALIZATION

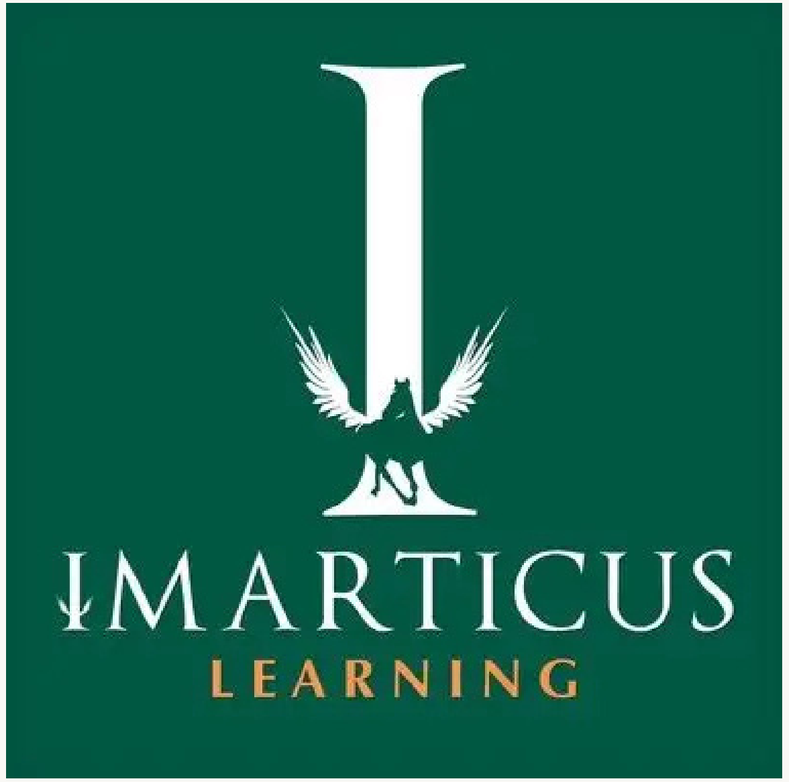






POWER BI VISUALLIZATION





THANK YOU