Hospital SQL Portfolio Assignment

SECTION A: Encounter Trends

Question 1 Total Encounters per Year

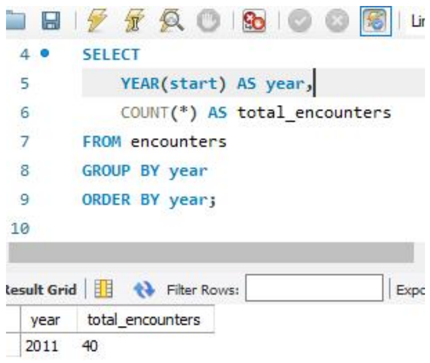
Explanation:

This query extracts the year from the encounter start date and counts how many encounters occurred in each year.

SQL Query:

```
SELECT
   YEAR(start) AS year,
   COUNT(*) AS total_encounters
FROM encounters
GROUP BY year
ORDER BY year;
```

Result:



```
SELECT
    YEAR(start) AS year,
    COUNT(*) AS total_encounters
FROM encounters
GROUP BY year
ORDER BY year;
```

```
year    total_encounters
2011    40
```

Question 2 Yearly % by Encounter Class

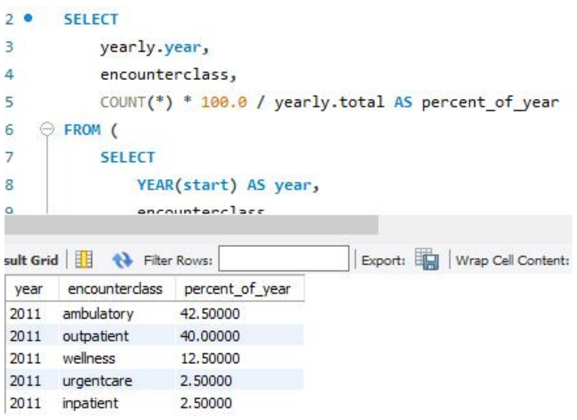
Explanation:

This query shows how the distribution of encounter types (like inpatient, emergency, outpatient) changes over time. It provides insight into which services are more commonly used each year.

SQL Query:

```
SELECT
  yearly.year,   encounterclass,
  COUNT(*) * 100.0 / yearly.total AS percent_of_year
FROM (
  SELECT    YEAR(start) AS year,
    encounterclass
  FROM encounters
) AS derived
JOIN (
  SELECT
    YEAR(start) AS year,
    COUNT(*) AS total
  FROM encounters
  GROUP BY YEAR(start)
) AS yearly ON derived.year = yearly.year
GROUP BY yearly.year, encounterclass
ORDER BY yearly.year, percent_of_year DESC;
```

Result :



```
SELECT
    yearly.year,
    encounterclass,
    COUNT(*) * 100.0 / yearly.total AS percent_of_year
FROM (
    SELECT
        YEAR(start) AS year,
        encounterclass
```

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

year	encounterclass	percent_of_year
2011	ambulatory	42.50000
2011	outpatient	40.00000
2011	wellness	12.50000
2011	urgentcare	2.50000
2011	inpatient	2.50000

Question 3 Duration-Based Classification

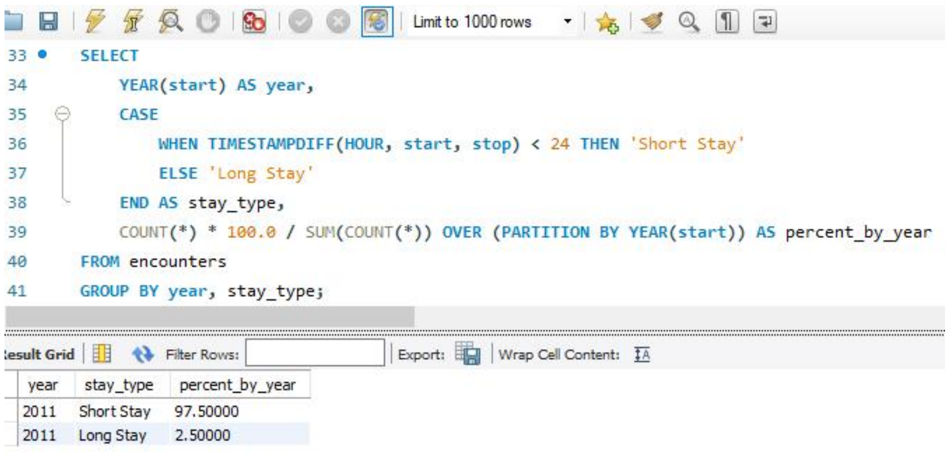
Explanation:

This query categorizes encounters into Short or Long stays based on whether they lasted less than 24 hours. It's useful for understanding how often the hospital handles short emergency visits versus extended admissions.

SQL Query:

```
SELECT
   YEAR(start) AS year,
   CASE
      WHEN TIMESTAMPDIFF(HOUR, start, stop) < 24 THEN 'Short Stay'
      ELSE 'Long Stay'
   END AS stay_type,
   COUNT(*) * 100.0 / SUM(COUNT(*)) OVER (PARTITION BY YEAR(start)) AS
percent_by_year
FROM encounters
GROUP BY year, stay_type;
```

Result:



```
SELECT
    YEAR(start) AS year,
    CASE
        WHEN TIMESTAMPDIFF(HOUR, start, stop) < 24 THEN 'Short Stay'
        ELSE 'Long Stay'
    END AS stay_type,
    COUNT(*) * 100.0 / SUM(COUNT(*)) OVER (PARTITION BY YEAR(start)) AS percent_by_year
FROM encounters
GROUP BY year, stay_type;
```

```
year  stay_type   percent_by_year
2011  Short Stay  97.50000
2011  Long Stay   2.50000
```

SECTION B: Financial & Coverage Insights

Question 4 Zero Payer Coverage

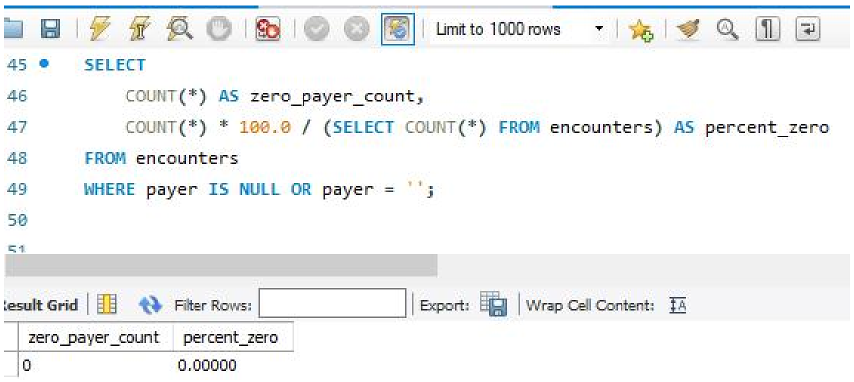
Explanation:

This identifies the number and percentage of encounters with no payer listed.

SQL Query:

```
SELECT
   COUNT(*) AS zero_payer_count,
   COUNT(*) * 100.0 / (SELECT COUNT(*) FROM encounters) AS percent_zero
FROM encounters
WHERE payer IS NULL OR payer = '';
```

Result:



```
SELECT
    COUNT(*) AS zero_payer_count,
    COUNT(*) * 100.0 / (SELECT COUNT(*) FROM encounters) AS percent_zero
FROM encounters
WHERE payer IS NULL OR payer = '';
```

zero_payer_count	percent_zero
0	0.00000

Question 5 Top 10 Frequent Procedures

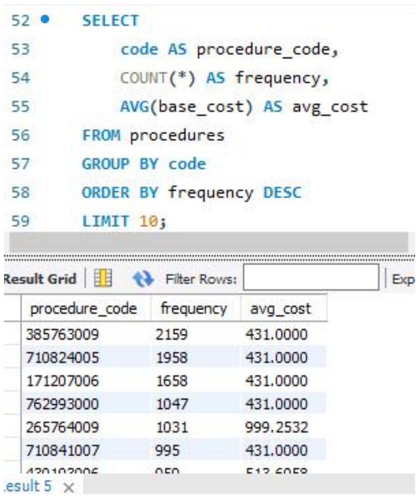
Explanation:

This query identifies the most commonly performed procedures and their average base cost. It helps the hospital understand which treatments are performed most often and plan resource allocation accordingly.

SQL Query:

```
SELECT
   code AS procedure_code,
   COUNT(*) AS frequency,
   AVG(base_cost) AS avg_cost
FROM procedures
GROUP BY code
ORDER BY frequency DESC
LIMIT 10;
```

Result :



```
SELECT
    code AS procedure_code,
    COUNT(*) AS frequency,
    AVG(base_cost) AS avg_cost
FROM procedures
GROUP BY code
ORDER BY frequency DESC
LIMIT 10;
```

procedure_code	frequency	avg_cost
385763009	2159	431.0000
710824005	1958	431.0000
171207006	1658	431.0000
762993000	1047	431.0000
265764009	1031	999.2532
710841007	995	431.0000
[illegible]	[illegible]	[illegible]

Question 6 Costliest Procedures

Explanation:

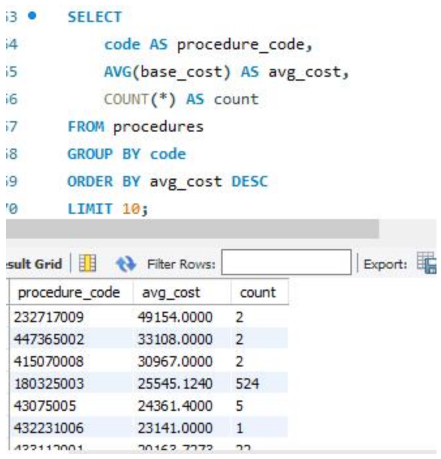
This query lists the 10 procedures with the highest average base cost and how many times each was performed. It provides insight into where the hospital incurs the most procedural expenses.

SQL Query:

```
SELECT
   code AS procedure_code,
   AVG(base_cost) AS avg_cost,
   COUNT(*) AS count
FROM procedures
GROUP BY code
ORDER BY avg_cost DESC
```

LIMIT 10;

Result:



```
SELECT
    code AS procedure_code,
    AVG(base_cost) AS avg_cost,
    COUNT(*) AS count
FROM procedures
GROUP BY code
ORDER BY avg_cost DESC
LIMIT 10;
```

procedure_code	avg_cost	count
232717009	49154.0000	2
447365002	33108.0000	2
415070008	30967.0000	2
180325003	25545.1240	524
43075005	24361.4000	5
432231006	23141.0000	1
[illegible]	[illegible]	[illegible]

Question 7 Claim Cost by Payer

Explanation:

This query calculates the average claim cost per payer.

SQL Query:

SELECT

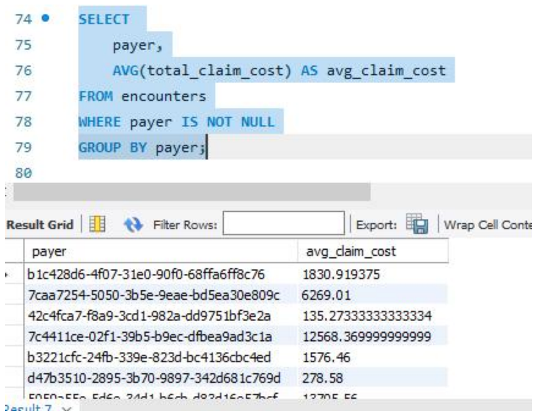
payer,

AVG(total_claim_cost) AS avg_claim_cost

FROM encounters

WHERE payer IS NOT NULL

GROUP BY payer;

Result: 

```
SELECT
    payer,
    AVG(total_claim_cost) AS avg_claim_cost
FROM encounters
WHERE payer IS NOT NULL
GROUP BY payer;
```

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

payer	avg_claim_cost
b1c428d6-4f07-31e0-90f0-68ffa6ff8c76	1830.919375
7caa7254-5050-3b5e-9eae-bd5ea30e809c	6269.01
42c4fca7-f8a9-3cd1-982a-dd9751bf3e2a	135.27333333333334
7c4411ce-02f1-39b5-b9ec-dfbea9ad3c1a	12568.369999999999
b3221cfc-24fb-339e-823d-bc4136cbc4ed	1576.46
d47b3510-2895-3b70-9897-342d681c769d	278.58

Result 7

SECTION C: Patient Behavior & Risk Analysis

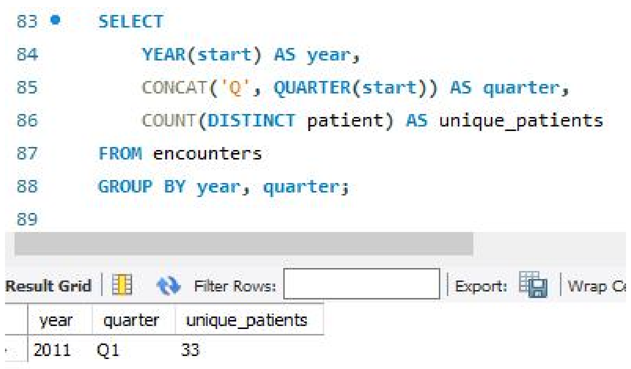
Question 8 Unique Patients per Quarter

Explanation:

This query calculates how many unique patients were admitted each quarter of each year. It helps uncover seasonal trends or spikes in admissions

SQL Query:

```
SELECT
   YEAR(start) AS year,
   CONCAT('Q', QUARTER(start)) AS quarter,
   COUNT(DISTINCT patient) AS unique_patients
FROM encounters
GROUP BY year, quarter;
```



```
SELECT
    YEAR(start) AS year,
    CONCAT('Q', QUARTER(start)) AS quarter,
    COUNT(DISTINCT patient) AS unique_patients
FROM encounters
GROUP BY year, quarter;
```

```
year   quarter   unique_patients
2011   Q1        33
```

Result:

Question 9 Readmissions within 30 Days

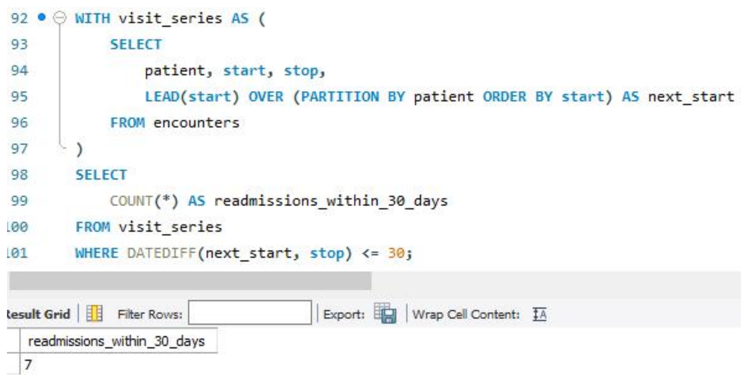
Explanation:

This detects how many patients were readmitted within 30 days of their last visit.

SQL Query:

```
WITH visit_series AS (
   SELECT
      patient, start, stop,
      LEAD(start) OVER (PARTITION BY patient ORDER BY start) AS next_start
   FROM encounters
)
SELECT
   COUNT(*) AS readmissions_within_30_days
FROM visit_series
WHERE DATEDIFF(next_start, stop) <= 30;
```

Result:



```
WITH visit_series AS (
    SELECT
        patient, start, stop,
        LEAD(start) OVER (PARTITION BY patient ORDER BY start) AS next_start
    FROM encounters
)
SELECT
    COUNT(*) AS readmissions_within_30_days
FROM visit_series
WHERE DATEDIFF(next_start, stop) <= 30;
```

```
readmissions_within_30_days
7
```

Question 10 Top 5 Most Readmitted Patients

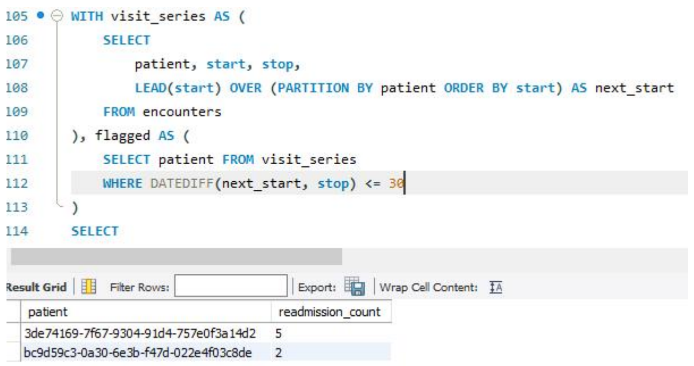
Explanation:

This builds on the previous query to find which individual patients had the highest number of readmissions. It's important for care teams to identify frequent returnees and manage their cases better.

SQL Query:

```
WITH visit_series AS (

  SELECT

    patient, start, stop,

    LEAD(start) OVER (PARTITION BY patient ORDER BY start) AS next_start

  FROM encounters

), flagged AS (

  SELECT patient FROM visit_series

  WHERE DATEDIFF(next_start, stop) <= 30

)

SELECT

  patient,

  COUNT(*) AS readmission_count

FROM flagged
```

```
GROUP BY patient

ORDER BY readmission_count DESC

LIMIT 5;
```



```
WITH visit_series AS (
    SELECT
        patient, start, stop,
        LEAD(start) OVER (PARTITION BY patient ORDER BY start) AS next_start
    FROM encounters
), flagged AS (
    SELECT patient FROM visit_series
    WHERE DATEDIFF(next_start, stop) <= 30
)
SELECT
```

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

patient	readmission_count
3de74169-7f67-9304-91d4-757e0f3a14d2	5
bc9d59c3-0a30-6e3b-f47d-022e4f03c8de	2

Result:

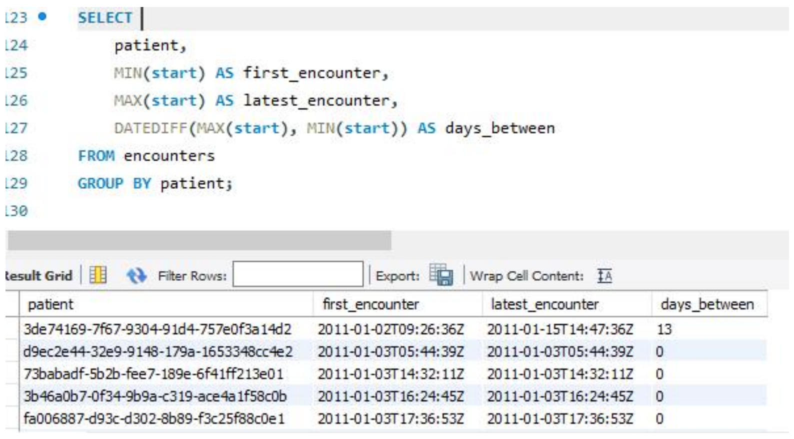
Question 11 First vs. Latest Encounter Analysis

Explanation:

This provides the time span between a patient's first and latest visit.

SQL Query:

```
SELECT

   patient,

   MIN(start) AS first_encounter,

   MAX(start) AS latest_encounter,

   DATEDIFF(MAX(start), MIN(start)) AS days_between

FROM encounters

GROUP BY patient;
```



```
SELECT
    patient,
    MIN(start) AS first_encounter,
    MAX(start) AS latest_encounter,
    DATEDIFF(MAX(start), MIN(start)) AS days_between
FROM encounters
GROUP BY patient;
```

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

patient	first_encounter	latest_encounter	days_between
3de74169-7f67-9304-91d4-757e0f3a14d2	2011-01-02T09:26:36Z	2011-01-15T14:47:36Z	13
d9ec2e44-32e9-9148-179a-1653348cc4e2	2011-01-03T05:44:39Z	2011-01-03T05:44:39Z	0
73babadf-5b2b-fee7-189e-6f41ff213e01	2011-01-03T14:32:11Z	2011-01-03T14:32:11Z	0
3b46a0b7-0f34-9b9a-c319-ace4a1f58c0b	2011-01-03T16:24:45Z	2011-01-03T16:24:45Z	0
fa006887-d93c-d302-8b89-f3c25f88c0e1	2011-01-03T17:36:53Z	2011-01-03T17:36:53Z	0

Result:

SECTION D: Advanced Logic

Question 12 CTE + CASE Pivot Table

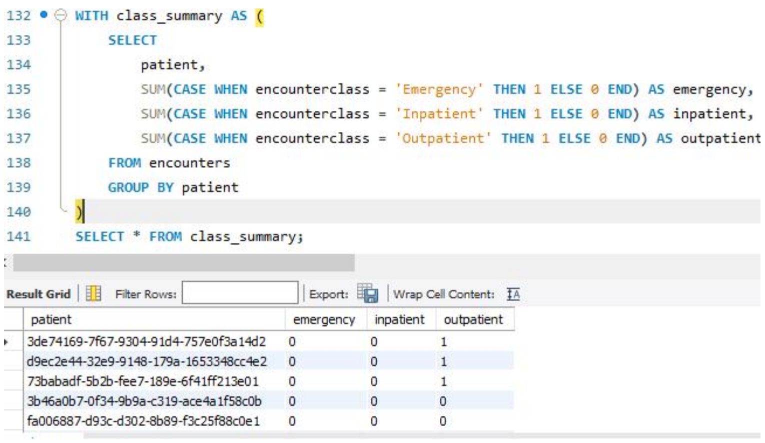
Explanation:

This query creates a pivot-style table showing how many encounters each patient had in each encounter class (e.g., Emergency, Inpatient, Outpatient). It gives a profile of patient interaction types

SQL Query:

```
WITH class_summary AS (
  SELECT
    patient,
    SUM(CASE WHEN encounterclass = 'Emergency' THEN 1 ELSE 0 END) AS emergency,
    SUM(CASE WHEN encounterclass = 'Inpatient' THEN 1 ELSE 0 END) AS inpatient,
    SUM(CASE WHEN encounterclass = 'Outpatient' THEN 1 ELSE 0 END) AS outpatient
  FROM encounters
  GROUP BY patient
)
SELECT * FROM class_summary;
```

Result:



```
WITH class_summary AS (
    SELECT
        patient,
        SUM(CASE WHEN encounterclass = 'Emergency' THEN 1 ELSE 0 END) AS emergency,
        SUM(CASE WHEN encounterclass = 'Inpatient' THEN 1 ELSE 0 END) AS inpatient,
        SUM(CASE WHEN encounterclass = 'Outpatient' THEN 1 ELSE 0 END) AS outpatient
    FROM encounters
    GROUP BY patient
)
SELECT * FROM class_summary;
```

patient	emergency	inpatient	outpatient
3de74169-7f67-9304-91d4-757e0f3a14d2	0	0	1
d9ec2e44-32e9-9148-179a-1653348cc4e2	0	0	1
73babadf-5b2b-fee7-189e-6f41ff213e01	0	0	1
3b46a0b7-0f34-9b9a-c319-ace4a1f58c0b	0	0	0
fa006887-d93c-d302-8b89-f3c25f88c0e1	0	0	0

Question 13 Most Recent Encounter per Patient

Explanation:

This query fetches the most recent encounter for every patient, along with any procedure that was performed. It's used to summarize the last known hospital interaction for each individual.

SQL Query:

WITH latest_visits AS (

SELECT patient, MAX(start) AS latest_start

FROM encounters

GROUP BY patient

)

SELECT

e.patient,

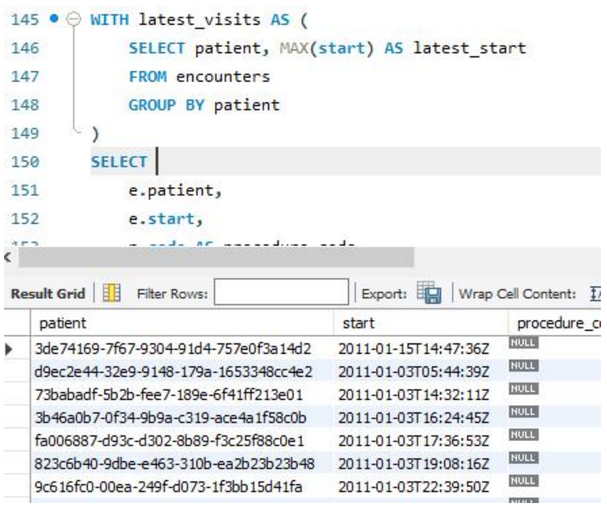
e.start,

p.code AS procedure_code

FROM encounters e

LEFT JOIN procedures p ON e.`ï»¿id` = p.code

```
JOIN latest_visits lv ON e.patient = lv.patient AND e.start = lv.latest_start;
```

Result: 

```
WITH latest_visits AS (
    SELECT patient, MAX(start) AS latest_start
    FROM encounters
    GROUP BY patient
)
SELECT
    e.patient,
    e.start,
```

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

	patient	start	procedure_c
▶	3de74169-7f67-9304-91d4-757e0f3a14d2	2011-01-15T14:47:36Z	NULL
	d9ec2e44-32e9-9148-179a-1653348cc4e2	2011-01-03T05:44:39Z	NULL
	73babadf-5b2b-fee7-189e-6f41ff213e01	2011-01-03T14:32:11Z	NULL
	3b46a0b7-0f34-9b9a-c319-ace4a1f58c0b	2011-01-03T16:24:45Z	NULL
	fa006887-d93c-d302-8b89-f3c25f88c0e1	2011-01-03T17:36:53Z	NULL
	823c6b40-9dbe-e463-310b-ea2b23b23b48	2011-01-03T19:08:16Z	NULL
	9c616fc0-00ea-249f-d073-1f3bb15d41fa	2011-01-03T22:39:50Z	NULL

Question 14 Top Diagnoses Per Age Group

Explanation:

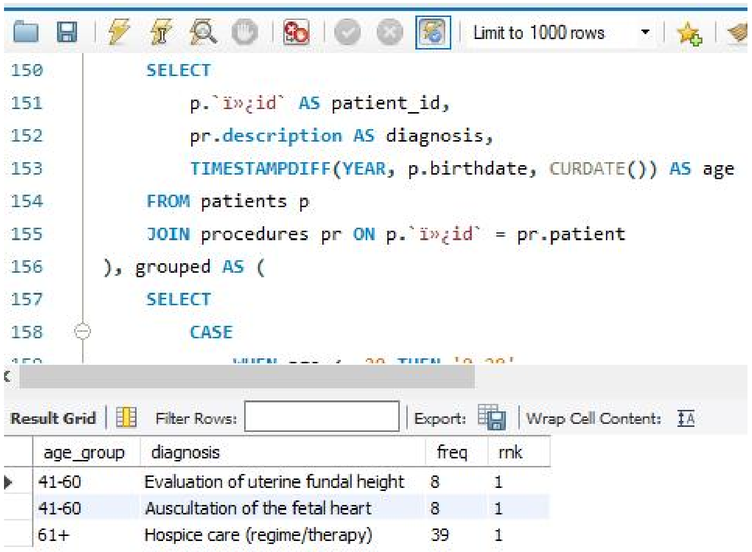
This query analyzes which diagnoses are most common in each age group. It helps the hospital focus on age-targeted healthcare programs and spot trends across generations.

SQL Query:

```
WITH age_diag AS (
  SELECT
    p.`ï»¿id` AS patient_id,
    pr.description AS diagnosis,
    TIMESTAMPDIFF(YEAR, p.birthdate, CURDATE()) AS age
  FROM patients p
  JOIN procedures pr ON p.`ï»¿id` = pr.patient
), grouped AS (
  SELECT
    CASE
```

```
            WHEN age <= 20 THEN '0-20'
            WHEN age <= 40 THEN '21-40'
            WHEN age <= 60 THEN '41-60'
            ELSE '61+'
        END AS age_group,
        diagnosis,
        COUNT(*) AS freq
    FROM age_diag
    GROUP BY age_group, diagnosis
), ranked AS (
    SELECT *, RANK() OVER (PARTITION BY age_group ORDER BY freq DESC) AS rnk
    FROM grouped
)
SELECT * FROM ranked WHERE rnk = 1;
```

Result :



Limit to 1000 rows

```
    SELECT
        p.`ï»¿id` AS patient_id,
        pr.description AS diagnosis,
        TIMESTAMPDIFF(YEAR, p.birthdate, CURDATE()) AS age
    FROM patients p
    JOIN procedures pr ON p.`ï»¿id` = pr.patient
), grouped AS (
    SELECT
        CASE
```

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

age_group	diagnosis	freq	rnk
41-60	Evaluation of uterine fundal height	8	1
41-60	Auscultation of the fetal heart	8	1
61+	Hospice care (regime/therapy)	39	1