Section A: Theory Questions

1a. Primary Key vs Candidate Key

Question: State the difference between Primary key and candidate key

Answer: A candidate key is any attribute or set of attributes that uniquely identifies each row in a table and can potentially serve as the primary key. A primary key is one specific candidate key chosen to enforce uniqueness and is used as the main identifier for the table, preventing NULL values and duplicate entries.

1b. String Functions

Question: Explain any two string functions with an example

Answer: CONCAT() combines multiple strings into one (e.g., SELECT CONCAT(FirstName, ' ', LastName) AS FullName FROM customers;). SUBSTRING() extracts a portion of a string (e.g., SELECT SUBSTRING('Hello World', 1, 5) AS Result; returns 'Hello').

1c. Aggregate Functions in WHERE

Question: Can we use aggregate functions in where clause? Justify your answer

Answer: No, aggregate functions like COUNT(), SUM() cannot be used directly in WHERE because WHERE filters rows before grouping. Use HAVING clause instead for post-grouping filters (e.g., SELECT department, COUNT*() FROM employees GROUP BY department HAVING COUNT()* > 5).

1d. Cross-Join Definition

Question: What is Cross-Join?

Answer: CROSS JOIN produces a Cartesian product of two tables, combining every row from the first table with every row from the second, resulting in rows × columns total rows without any matching condition.

1e. Filtering GROUP BY Results

Question: Aryan, a database administrator, has grouped records of a table with the help of group by clause. He needs to further filter groups of records generated through group by clause. Suggest suitable clause for it and properly explain its usage with the help of an example

Answer: Use HAVING clause. HAVING filters aggregated groups after GROUP BY (e.g., SELECT city, AVG(salary) FROM employees GROUP BY city HAVING AVG(salary) > 50000;) while WHERE filters individual rows before grouping.

2a. Window Functions Uses

Question: What are the uses of Window Function?

Answer: Window functions perform calculations across row sets related to the current row without collapsing rows (uses: ranking with ROW_NUMBER(), running totals with SUM() OVER(), percentiles with PERCENT_RANK()).

2b. EXISTS Operator

Question: Explain about EXISTS operator

Answer: EXISTS tests if a subquery returns any rows, returning TRUE if at least one row exists (efficient for checking existence; e.g., SELECT * FROM customers WHERE EXISTS (SELECT 1 FROM orders WHERE orders.customer_id = customers.id)).

2c. Percent Rank

Question: What is percent rank? Explain with an example

Answer: PERCENT_RANK() returns relative rank as a value between 0 and 1 within a window (e.g., SELECT name, salary, PERCENT_RANK() OVER (ORDER BY salary) FROM employees; shows 0 for lowest, 1 for highest salary percentile).

2d. Database Locks Types

Question: State and explain in brief types of Locks in database

Answer: Shared Lock (allows multiple reads, blocks writes), Exclusive Lock (blocks all access for writes), Update Lock (allows reads, converts to exclusive for updates) manage concurrent access.

2e. Column Data Restrictions

Question: Explain the different ways you can restrict the data in a column. Demonstrate with Examples

Answer: CHECK constraint (CHECK (age >= 18)), NOT NULL, DEFAULT (age INT DEFAULT 0), FOREIGN KEY, UNIQUE enforce restrictions.

Section B: Hospital Database Queries

Note: Using hospital database with worldhappinessreport table

3a. Lowest Corruption Perception

Question: From the worldhappinessreport table, identify the country where the perception of corruption was the lowest in the most recent year available

```sql
SELECT Countryname
FROM worldhappinessreport
WHERE Perceptionsofcorruption = (
    SELECT MIN(Perceptionsofcorruption)
    FROM worldhappinessreport
    WHERE year = (SELECT MAX(year) FROM worldhappinessreport)
);
```

3b. Top 5 Social Support by Year

Question: List the top 5 countries with the highest social support scores for every year. Arrange the data in ascending order of year

```sql
SELECT year, Countryname, Socialsupport
FROM (
    SELECT year, Countryname, Socialsupport,
           ROW_NUMBER() OVER (PARTITION BY year ORDER BY
Socialsupport DESC) as rn
    FROM worldhappinessreport
) ranked
WHERE rn <= 5
ORDER BY year ASC;
```

3c. High Freedom Countries 2019-2020

Question: Identify countries where the freedom to make life choices was greater than 0.8 for both 2019 and 2020

```sql
SELECT DISTINCT w1.Countryname
FROM worldhappinessreport w1
JOIN worldhappinessreport w2 ON w1.Countryname = w2.Countryname
WHERE w1.year = 2019 AND w2.year = 2020
  AND w1.Freedomtomakelifechoices > 0.8
  AND w2.Freedomtomakelifechoices > 0.8;
```

3d. Yearly NegativeAffect Average

Question: Calculate the yearly average of NegativeAffect across all countries and arrange the data from lowest to highest

```sql
SELECT year, AVG(Negativeaffect) as avg_negative_affect
FROM worldhappinessreport
GROUP BY year
ORDER BY avg_negative_affect ASC;
```

3e. Highest Healthy Life Expectancy

Question: Retrieve the country with the highest Healthylifeexpectancyatbirth and its corresponding value

```
SELECT Countryname, Healthylifeexpectancyatbirth
FROM worldhappinessreport
WHERE Healthylifeexpectancyatbirth = (
    SELECT MAX(Healthylifeexpectancyatbirth) FROM
worldhappinessreport
);
```

3f. Rolling PositiveAffect Average

Question: Compute a rolling average of Positiveaffect for each country using a 3-year moving window

```
SELECT Countryname, year, Positiveaffect,
       AVG(Positiveaffect) OVER (
           PARTITION BY Countryname
           ORDER BY year
           ROWS BETWEEN 2 PRECEDING AND CURRENT ROW
       ) as rolling_avg
FROM worldhappinessreport
ORDER BY Countryname, year;
```

3g. Lowest Social Support Subquery

Question: Write a subquery to find the lowest Socialsupport value in the most recent year and identify the corresponding country

```
SELECT Countryname, Socialsupport
FROM worldhappinessreport
WHERE year = (SELECT MAX(year) FROM worldhappinessreport)
  AND Socialsupport = (
      SELECT MIN(Socialsupport)
      FROM worldhappinessreport
      WHERE year = (SELECT MAX(year) FROM worldhappinessreport)
  );
```

3h. Top Providers by Discharges

Question: Generate a report to list healthcare providers with the highest number of discharges, including the average charges and payments per provider, sorted in descending order

```
SELECT ProviderName,
       SUM(TotalDischarges) as total_discharges,
       AVG(CAST(AverageCoveredCharges AS DECIMAL(10,2))) as
avg_covered_charges,
       AVG(CAST(AverageTotalPayments AS DECIMAL(10,2))) as
avg_total_payments
FROM inpatient1
```

```
GROUP BY ProviderName
ORDER BY total_discharges DESC;
```

3i. Insurance Coverage Ratio

Question: Generate a report displaying the insurance coverage ratio for all healthcare providers, arranged from lowest to highest. The insurance coverage ratio is calculated as AverageCoveredCharges ÷ AverageTotalPayments ÷ AverageMedicarePayments

```
SELECT ProviderName,
       (CAST(AverageCoveredCharges AS DECIMAL(10,2)) /
        CAST(AverageTotalPayments AS DECIMAL(10,2)) /
        CAST(AverageMedicarePayments AS DECIMAL(10,2))) as
coverage_ratio
FROM inpatient1
ORDER BY coverage_ratio ASC;
```

Section C: Sales Database Queries

Note: Using sales schema (schema structure in sales_schema.sql)

4a. Customers with Similar First Names

Question: Write a Query to display the customers who have similar first name. Use Tables customer

```
SELECT DISTINCT c1.CustomerId, c1.FirstName, c1.LastName
FROM customer c1
JOIN customer c2 ON c1.FirstName = c2.FirstName
                AND c1.CustomerId != c2.CustomerId
ORDER BY c1.FirstName;
```

4b. Previous Invoice Dates

Question: Write a Query to find the customerid, invoice date and previous invoice dates for every customer. Use Tables invoice

```
SELECT i1.CustomerId,
       i1.InvoiceDate,
       LAG(i1.InvoiceDate) OVER (PARTITION BY i1.CustomerId ORDER
BY i1.InvoiceDate) as previous_invoice_date
FROM invoice i1
ORDER BY i1.CustomerId, i1.InvoiceDate;
```

4c. Tracks by Genre Price Filter

Question: Write a Query to find the total tracks for each genre, which has unit price more than 0.9 and genre name starts with s. Use Tables track, genre

```sql
SELECT g.Name, COUNT(t.TrackId) as total_tracks
FROM track t
JOIN genre g ON t.GenreId = g.GenreId
WHERE t.UnitPrice > 0.9 AND g.Name LIKE 's%'
GROUP BY g.GenreId, g.Name;
```

4d. Customer Album Ranking

Question: Using the sales database, write a query to rank customers based on number of unique albums purchased by them in the lifetime. Use Tables track, invoiceline, invoice

```sql
SELECT i.CustomerId,
       COUNT(DISTINCT t.AlbumId) as unique_albums,
       RANK() OVER (ORDER BY COUNT(DISTINCT t.AlbumId) DESC) as
album_rank
FROM invoice i
JOIN invoiceline il ON i.InvoiceId = il.InvoiceId
JOIN track t ON il.TrackId = t.TrackId
GROUP BY i.CustomerId
ORDER BY unique_albums DESC;
```

4e. Tracks per Composer

Question: Write a Query to get the details of total number of tracks billed per composer. Sort the result in the descending order of the total tracks billed. Use Tables track, invoiceline

```sql
SELECT t.Composer, COUNT(il.InvoiceLineId) as total_tracks_billed
FROM track t
JOIN invoiceline il ON t.TrackId = il.TrackId
WHERE t.Composer IS NOT NULL
GROUP BY t.Composer
ORDER BY total_tracks_billed DESC;
```

4f. Virtual Table Albums per Order

Question: Using the sales database, write a query to create a virtual table to display the details like customerid, billing city and customer city and the number of albums billed in each order placed/invoice raised. Use Tables track, invoiceline, invoice, customer

```sql
WITH album_counts AS (
    SELECT i.CustomerId,
           i.BillingCity,
           c.City as CustomerCity,
           i.InvoiceId,
           COUNT(DISTINCT t.AlbumId) as albums_billed
    FROM invoice i
    JOIN customer c ON i.CustomerId = c.CustomerId
    JOIN invoiceline il ON i.InvoiceId = il.InvoiceId
    JOIN track t ON il.TrackId = t.TrackId
```

```sql
    GROUP BY i.CustomerId, i.BillingCity, c.City, i.InvoiceId
)
SELECT CustomerId, BillingCity, CustomerCity, albums_billed
FROM album_counts
ORDER BY CustomerId, InvoiceId;
```