Analytics Module — Migration Map (HTTP Dashboard → GraphQL)

Objective: Provide exact file paths and copy■pasteable code to integrate an isolated Analytics module in API■GATEWAY, replicating the old HTTP Dashboard outputs with minimal changes. Paste each snippet into a new file at the given path, then apply the listed modifications to existing files.

1) NEW FILES (create these paths and paste code)

Path: `src/analytics/analytics.config.ts`

```
import { registerAs } from '@nestjs/config';

/**
 * Analytics configuration (isolated from live-mode and historical-mode).
 * Reuses the legacy aggregator backend while allowing environment-based limits.
 */
export const analyticsConfig = registerAs('analytics', () => ({
  // Aggregator base URL (same backend the old HTTP Dashboard used):
  recordsBaseURL:
    process.env.CALL_METRICS_RECORDS_URL ?? 'http://localhost:3001',

  // Limits and timeouts (tweak via env if desired):
  maxPageSize: Number(process.env.ANALYTICS_MAX_PAGE_SIZE ?? 200),
  maxRangeDays: Number(process.env.ANALYTICS_MAX_RANGE_DAYS ?? 366),
  requestTimeoutMs: Number(process.env.ANALYTICS_REQUEST_TIMEOUT_MS ?? 15000),
}));
```

Path: `src/analytics/analytics.module.ts`

```
import { Module } from '@nestjs/common';
import { ConfigModule } from '@nestjs/config';
import { analyticsConfig } from './analytics.config';

// Calls Duration Summary (first feature)
import { CallsDurationSummaryResolver } from './summaries/calls-duration-summary/calls-duration-summary.reso
import { CallsDurationSummaryService } from './summaries/calls-duration-summary/calls-duration-summary.servic

@Module({
  imports: [ConfigModule.forFeature(analyticsConfig)],
  providers: [
    CallsDurationSummaryResolver,
    CallsDurationSummaryService,
    // Add more resolvers/services here as you migrate other Analytics features.
  ],
})
export class AnalyticsModule {}
```

Path: `src/analytics/common/dto/range.graphql`

```
scalar DateTime

input Range {
  startDate: DateTime!
  endDate: DateTime!
}
```

Path: `src/analytics/summaries/calls-duration-summary/calls-duration-summary.graphql`

```
# Calls Duration Summary (matches legacy HTTP Dashboard semantics)
type CallsDurationSummary {
  totalAiOperationTime: Int!
  totalAiNonOperationTime: Int!
  totalHumanAgentTime: Int!
  unit: String!  # "seconds"
}

extend type Query {
```

```
  callsDurationSummary(range: Range!, clinicTimezone: String!): CallsDurationSummary!
}
```

Path: src/analytics/utils/stringify-pipeline.util.ts

```
export function stringifyPipeline(pipeline: unknown[]): string {
  const encoded = encodeURIComponent(JSON.stringify(pipeline));
  return `pipeline=${encoded}`;
}
```

Path: src/analytics/utils/http.util.ts

```
export async function httpGet(url: string, timeoutMs = 15000, headers: Record<string,string> = {}) {
  const controller = new AbortController();
  const timer = setTimeout(() => controller.abort(), timeoutMs);
  try {
    const res = await fetch(url, { method: 'GET', headers, signal: controller.signal });
    if (!res.ok) {
      const text = await res.text().catch(() => '');
      throw new Error(`UPSTREAM_ERROR ${res.status}: ${text}`);
    }
    return res.json();
  } finally {
    clearTimeout(timer);
  }
}
```

Path: src/analytics/summaries/calls-duration-summary/calls-duration-summary.pipeline.ts

```
type Range = { startDate: string; endDate: string };

export function buildCallsDurationSummaryPipeline(args: {
  range: Range;
  clinicTimezone: string;
}) {
  const { range } = args;
  return [
    {
      $match: {
        $expr: {
          $and: [
            { $gte: ["$callStartTime", { $toDate: range.startDate }] },
            { $lte: ["$callStartTime", { $toDate: range.endDate }] },
          ],
        },
      },
    },
    {
      $project: {
        outcome: 1,
        callStartTime: 1,
        callEndTime: 1,
        durationSeconds: {
          $divide: [
            { $subtract: [{ $toDate: "$callEndTime" }, { $toDate: "$callStartTime" }] },
            1000,
          ],
        },
      },
    },
    {
      $group: {
        _id: null,
        totalAiOperationTime: {
          $sum: {
            $cond: [
              { $in: ["$outcome", ["scheduled", "rescheduled", "cancellation"]] },
              "$durationSeconds",
              0,
            ],
          },
        },
        totalAiNonOperationTime: {
          $sum: {
            $cond: [{ $eq: ["$outcome", "not-available"] }, "$durationSeconds", 0],
```

```
            },
          },
          totalHumanAgentTime: {
            $sum: {
              $cond: [{ $eq: ["$outcome", "handled-by-agent"] }, "$durationSeconds", 0],
            },
          },
        },
      },
      {
        $project: {
          _id: 0,
          totalAiOperationTime: { $ifNull: ["$totalAiOperationTime", 0] },
          totalAiNonOperationTime: { $ifNull: ["$totalAiNonOperationTime", 0] },
          totalHumanAgentTime: { $ifNull: ["$totalHumanAgentTime", 0] },
          unit: { $literal: "seconds" },
        },
      },
    ];
  }
```

Path:
src/analytics/summaries/calls-duration-summary/calls-duration-summary.service.ts

```
import { Inject, Injectable } from '@nestjs/common';
import { ConfigType } from '@nestjs/config';
import { analyticsConfig } from '../../analytics.config';
import { buildCallsDurationSummaryPipeline } from './calls-duration-summary.pipeline';
import { stringifyPipeline } from '../../../analytics/utils/stringify-pipeline.util';
import { httpGet } from '../../../analytics/utils/http.util';

@Injectable()
export class CallsDurationSummaryService {
  constructor(
    @Inject(analyticsConfig.KEY)
    private readonly cfg: ConfigType<typeof analyticsConfig>,
  ) {}

  async getSummary(range: { startDate: string; endDate: string }, clinicTimezone: string) {
    if (!range?.startDate || !range?.endDate) {
      throw new Error('INPUT_INVALID: range.startDate and range.endDate are required');
    }
    if (new Date(range.startDate) >= new Date(range.endDate)) {
      throw new Error('INPUT_INVALID: startDate must be < endDate');
    }

    const pipeline = buildCallsDurationSummaryPipeline({ range, clinicTimezone });
    const qs = stringifyPipeline(pipeline);
    const url = `${this.cfg.recordsBaseURL}/call-records/aggregate?${qs}`;
    const raw = await httpGet(url, this.cfg.requestTimeoutMs);

    const first = Array.isArray(raw) ? raw[0] : raw;
    return {
      totalAiOperationTime: Math.round(first?.totalAiOperationTime ?? 0),
      totalAiNonOperationTime: Math.round(first?.totalAiNonOperationTime ?? 0),
      totalHumanAgentTime: Math.round(first?.totalHumanAgentTime ?? 0),
      unit: 'seconds',
    };
  }
}
```

Path:
src/analytics/summaries/calls-duration-summary/calls-duration-summary.resolver.ts

```
import { Args, Query, Resolver } from '@nestjs/graphql';
import { CallsDurationSummaryService } from './calls-duration-summary.service';

@Resolver()
export class CallsDurationSummaryResolver {
  constructor(private readonly service: CallsDurationSummaryService) {}

  @Query('callsDurationSummary')
  async callsDurationSummary(
    @Args('range') range: { startDate: string; endDate: string },
    @Args('clinicTimezone') clinicTimezone: string,
  ) {
```

```
    return this.service.getSummary(range, clinicTimezone);
  }
}
```

2) MODIFICATIONS TO EXISTING FILES

Path to modify: `src/app.module.ts`

Add the AnalyticsModule import and include it in the @Module imports array (minimal diff):

```
// + NEW import
import { AnalyticsModule } from './analytics/analytics.module';

@Module({
  imports: [
    // ... existing imports (GraphQLModule, IamModule, LiveModeModule, HistoricalModeModule, PubSubModule, e
    AnalyticsModule, // + add this line
  ],
  // providers, etc...
})
export class AppModule {}
```

GraphQL config: No changes needed (already using `typePaths: ['./**/*.graphql']`).

Types generation: run your existing generator after adding the new SDL files (e.g., `npm run gen-types`).

Optional `.env` entries:

```
ANALYTICS_MAX_PAGE_SIZE=200
ANALYTICS_MAX_RANGE_DAYS=366
ANALYTICS_REQUEST_TIMEOUT_MS=15000
```