100 R Programming Interview Questions and Answers

R Basics

1. What is R and why is it used in data analysis?

R is an open-source programming language mainly used for statistical computing and data visualization. It is widely adopted by data analysts for tasks such as data cleaning, exploration, analysis, and reporting. It has powerful libraries like dplyr, ggplot2, and shiny that help in end-to-end data workflows.

2. How do you assign a value to a variable in R?

In R, values can be assigned using either the <- operator or the = sign.

Example:

```
x <- 5
y = 10
```

The <- operator is preferred in R programming for clarity and consistency.

3. What are vectors in R?

A vector is a basic data structure in R that contains elements of the same type. It is created using the c() function.

Example:

```
numbers <- c(1, 2, 3, 4)
```

All elements in a vector must be of the same data type (numeric, character, etc.).

4. How do you check the data type of an object in R?

You can use the class() or typeof() function.

Example:

```
x <- 100
class(x)      # Output: "numeric"
typeof(x)     # Output: "double"
```

5. What are data frames in R and how are they created?

A data frame is a table or 2D array-like structure in which each column can contain different types of data. It is created using the data.frame() function.

Example:

df <- data.frame(Name = c("Amit", "Raj"), Age = c(25, 30))

Data frames are one of the most commonly used data structures for analysis in R.

6. How do you read a CSV file in R?

You can use the read.csv() function to load CSV data into a data frame.

Example:

data <- read.csv("filename.csv")

Make sure the working directory is correctly set using setwd() if needed.

7. What is the difference between a matrix and a data frame in R?

- A matrix can only contain one data type (numeric, character, etc.).
- A data frame can contain multiple data types across columns.

Example:

matrix <- matrix(1:6, nrow = 2)

df <- data.frame(Name = c("A", "B"), Age = c(21, 22))

8. How do you install and load a package in R?

To install a package, use:

install.packages("ggplot2")

To load it into your session:

library(ggplot2)

---

9. What function is used to get the structure of a data frame in R?

Use the str() function to get the internal structure of an object.

Example:

str(df)

This gives column names, data types, and a preview of the data.

---

10. How do you write comments in R?

Use the # symbol to add comments.

Example:

```
# This is a comment in R
x <- 5  # Assignig val
```

Data Manipulation

11. What is the dplyr package and why is it useful for data manipulation in R?

dplyr is a grammar of data manipulation in R that provides fast, consistent functions for working with data frames and tibbles. It simplifies common data tasks such as filtering, selecting, grouping, and summarising.

Key features:

- Chainable functions using the pipe (%>%)
- Readable syntax close to English
- Works efficiently with large datasets

12. What is the pipe operator %>% and how is it used in dplyr?

The pipe operator %>% allows you to write code in a readable, step-by-step way, where the result of one operation is passed as the input to the next.

Example:

```
library(dplyr)


data %>%
  filter(Age > 25) %>%
  select(Name, Age) %>%
  arrange(desc(Age))
```

This avoids nesting and improves readability.

13. What is the difference between filter() and select() in dplyr?

- filter() is used to subset rows based on conditions.
- select() is used to choose specific columns from a dataset.

Example:

```
filter(data, Age > 30)      # Rows where Age > 30
select(data, Name, Age)     # Only Name and Age columns
```

14. How does mutate() work in dplyr?

mutate() is used to create new columns or modify existing ones.

Example:

```
data %>% mutate(Salary_after_tax = Salary * 0.7)
```

It adds a new column called Salary_after_tax with values computed from the Salary column.

15. What is the use of group_by() and summarise() in dplyr?

These functions help with grouped calculations.

Example:

```
data %>%
  group_by(Department) %>%
  summarise(AverageSalary = mean(Salary, na.rm = TRUE))
```

This groups data by Department and calculates the average salary for each group.

16. How do you arrange rows in ascending or descending order using dplyr?

Use the arrange() function:

```
arrange(data, Age)              # Ascending order
arrange(data, desc(Salary))     # Descending order
```

You can also sort by multiple columns.

17. What is the difference between data.frame and tibble?

- data.frame is the default R data structure for tabular data.
- tibble is a modern version provided by the tibble package, used by dplyr.

Differences:

- Tibbles do not automatically convert strings to factors.
- Tibbles display better in the console.
- They support non-standard column names (e.g., with spaces).

18. How do you rename columns in a dataset using dplyr?

Use the rename() function:

data %>% rename(NewName = OldName)

It's often combined with other functions using pipes for clarity.

These functions are helpful during aggregation tasks.

---

19. How can you create conditional columns in R similar to SQL's CASE WHEN?

Use the ifelse() or case_when() functions.

Example using case_when():

```
data %>%
  mutate(Category = case_when(
    Age < 18 ~ "Minor",
    Age >= 18 & Age <= 60 ~ "Adult",
    TRUE ~ "Senior"
  ))
```

This creates a new column called Category based on the value of Age.

---

20. What is the use of n() and n_distinct() in dplyr?

- n() counts the number of rows in a group
- n_distinct() counts the number of unique values

Example:

```
data %>%
  group_by(Region) %>%
  summarise(Total = n(), UniqueUsers = n_distinct(UserID))
```

21. What is data.table and how does it differ from dplyr?

data.table is another R package used for fast and memory-efficient data manipulation. It uses a different syntax compared to dplyr.

Key differences:

- data.table is faster for large datasets.
- Syntax is more compact but less readable for beginners.
- data.table modifies data by reference, whereas dplyr often returns a new object.

---

22. How do you convert a data.frame to data.table and vice versa?

```
library(data.table)

dt <- as.data.table(df)    # Convert data.frame to data.table
df <- as.data.frame(dt)    # Convert back to data.frame
```

23. How do you perform filtering and aggregation in data.table?

```
library(data.table)
dt <- data.table(Sales = c(100, 200, 150), Region = c("East", "West", "East"))

dt[Region == "East", .(TotalSales = sum(Sales))]
```

This filters rows where Region is "East" and computes the total sales.

24. What is chaining in data.table and how is it used?

Chaining refers to running multiple operations one after the other using [].

Example:

```
dt[, .(Total = sum(Sales)), by = Region][order(-Total)]
```

Here, we aggregate and then sort using a single line.

25. How do you merge or join datasets in R using dplyr?

Use the family of *_join() functions:

- left_join()
- right_join()

- inner_join()
- full_join()

Example:

left_join(df1, df2, by = "ID")

These work similar to SQL joins and require a common key.

Data Cleaning In R

26. How do you identify missing values in R?

Missing values in R are represented by NA. You can identify them using:

- is.na() – Returns a logical vector indicating TRUE for missing entries.
- anyNA() – Checks if any missing values exist in a dataset.

Example:

```
anyNA(data)            # TRUE if any NA is present
sum(is.na(data$Age))   # Count of missing Age values
```

27. How do you remove rows with missing values in R?

Use the na.omit() function or drop_na() from the tidyr package.

Example:

```
cleaned_data <- na.omit(data)
```

Or:

```
library(tidyr)
cleaned_data <- drop_na(data)
```

This removes all rows with one or more missing values.

28. How do you replace or impute missing values in R?

There are multiple methods to impute:

- **Replace with mean/median**:

```
data$Age[is.na(data$Age)] <- mean(data$Age, na.rm = TRUE)
```

- **Use mutate() and ifelse()**:

```
data <- data %>%
  mutate(Age = ifelse(is.na(Age), median(Age, na.rm = TRUE), Age))
```

- **Advanced imputation**: Use mice, missForest, or imputeTS packages.

29. How do you detect and handle duplicate records in R?

- duplicated() – Identifies duplicate rows.
- distinct() – Removes duplicate rows using dplyr.

Example:

```
duplicates <- data[duplicated(data), ]
unique_data <- distinct(data)
```

You can also use data.table::unique() or data.table[!duplicated(data.table)].

---

30. How do you standardize column names in R for consistency?

To make all column names lowercase and replace spaces:

```
names(data) <- tolower(names(data))
names(data) <- gsub(" ", "_", names(data))
```

Or use janitor::clean_names() for more automated cleaning:

```
library(janitor)
data <- clean_names(data)
```

---

31. How do you detect and remove outliers in R?

Basic methods include:

- **Using IQR (Interquartile Range)**:

```
Q1 <- quantile(data$Income, 0.25)
Q3 <- quantile(data$Income, 0.75)
IQR <- Q3 - Q1

outliers <- data %>%
  filter(Income < (Q1 - 1.5 * IQR) | Income > (Q3 + 1.5 * IQR))
```

- **Using boxplot.stats()**:

```
boxplot.stats(data$Income)$out
```

---

32. How do you convert data types in R?

Use coercion functions such as:

- as.numeric()
- as.character()
- as.factor()
- as.Date()

Example:

data$Age <- as.numeric(data$Age)

data$Date <- as.Date(data$Date, format = "%Y-%m-%d")

This is crucial when importing raw data.

---

33. How do you handle inconsistent text values in R?

Use string manipulation functions:

- tolower() / toupper() – Standardize case
- trimws() – Remove extra spaces
- gsub() – Replace substrings

Example:

data$City <- tolower(trimws(data$City))

data$City <- gsub("new delhi", "delhi", data$City)

You can also use stringr functions for more advanced tasks.

---

34. How do you reshape or pivot data in R?
Use pivot_longer() and pivot_wider() from tidyr:

- **Wide to long**:

pivot_longer(data, cols = c(Jan:Dec), names_to = "Month", values_to = "Sales")

- **Long to wide**:

pivot_wider(data, names_from = Category, values_from = Sales)

These are similar to pivot/unpivot in SQL and Excel.

---

35. What is the difference between drop_na(), replace_na(), and fill() in tidyr?

- drop_na() – Removes rows with NA values.
- replace_na() – Replaces missing values with specified value.
- fill() – Fills missing values from previous or next rows.

Example:

```
data <- replace_na(data, list(Sales = 0))
```

```
data <- fill(data, .direction = "down")
```

Each of these is useful depending on the business logic and context.

Data Visualization In R

36. What is ggplot2 and why is it used in R for data visualization?

ggplot2 is a popular visualization package in R based on the Grammar of Graphics. It provides a consistent, layered approach to building plots by mapping aesthetics (like x, y, color) to data and adding graphical layers.

Advantages:

- Highly customizable and professional-looking plots
- Handles large datasets
- Supports complex visualizations like facets, themes, annotations

37. How do you create a basic bar chart using ggplot2?

```
library(ggplot2)
ggplot(data, aes(x = Category)) +
  geom_bar()
```

- This generates a bar chart showing the count of each category.
- Use geom_col() if you already have summary data with counts.

38. How do you add labels, titles, and themes to a ggplot chart?

Use the following functions:

- labs() – Add title, subtitle, x and y labels
- theme_minimal(), theme_bw() – Change background and styling

Example:

```
ggplot(data, aes(x = Product, y = Sales)) +
  geom_col(fill = "skyblue") +
  labs(title = "Sales by Product", x = "Product", y = "Sales") +
  theme_minimal()
```

39. How can you create a line plot in R?

Line plots are useful for time series and trends:

```
ggplot(data, aes(x = Date, y = Revenue)) +
geom_line(color = "blue")
```

Use scale_x_date() to format the x-axis if needed.

40. What is faceting in ggplot2 and how is it used?

Faceting creates subplots for each category of a variable:

```
ggplot(data, aes(x = Month, y = Sales)) +
  geom_col() +
  facet_wrap(~ Region)
```

- facet_wrap() – Wraps plots by a single variable
- facet_grid() – Creates a matrix using two variables

41. How do you create a histogram in R using ggplot2?

```
ggplot(data, aes(x = Age)) +
  geom_histogram(binwidth = 5, fill = "orange")
```

Histograms show the distribution of a numeric variable by grouping into bins.

42. How do you customize colors and legends in a ggplot2 plot?

- Use fill = or color = inside aes() to map to variables
- Use scale_fill_manual() or scale_color_manual() to define custom colors

Example:

```
ggplot(data, aes(x = Category, fill = Region)) +
  geom_bar(position = "dodge") +
  scale_fill_manual(values = c("blue", "green", "red"))
```

43. What is the difference between geom_point(), geom_line(), and geom_smooth()?

- geom_point() – For scatter plots
- geom_line() – For continuous lines (usually time series)
- geom_smooth() – For adding trend lines (e.g., linear regression)

You can combine them:

```
ggplot(data, aes(x = Time, y = Sales)) +
  geom_point() +
  geom_smooth(method = "lm")
```

44. How do you make interactive plots in R?

Use the plotly package to convert static ggplot2 plots into interactive ones:

```
library(plotly)
p <- ggplot(data, aes(x = Product, y = Sales)) + geom_col()
ggplotly(p)
```

You can also use plot_ly() directly for fully interactive dashboards.

45. What are some best practices for data visualization in R?

- Always label axes and add a title using labs()
- Choose the right type of chart for the data (bar for categories, line for trends)
- Avoid clutter – limit use of too many colors or dimensions
- Use consistent themes (theme_minimal(), theme_light())
- Highlight insights, not just display data

Good visualizations help in storytelling and decision-making.

Statistical Analysis In R

46. How do you perform a correlation analysis in R?

To measure the relationship between two numeric variables, use:

cor(data$Variable1, data$Variable2, use = "complete.obs")

- Use "complete.obs" to handle missing values
- You can also generate a correlation matrix for all numeric columns using:

cor(data, use = "complete.obs")

47. How do you run a linear regression model in R?

Use the lm() function:

model <- lm(Sales ~ Marketing_Spend, data = data)

summary(model)

- The formula format is response ~ predictor
- summary() provides coefficients, R-squared, and p-values

48. How do you check if variables are statistically significant in a regression model?

In the output of summary(model), check:

- **p-values**: If p-value < 0.05, the variable is considered statistically significant
- **t-value**: Higher t-value indicates stronger contribution

Always interpret significance in the context of business use cases.

49. How do you perform hypothesis testing in R?

Use statistical test functions like:

- t.test() – For comparing means
- chisq.test() – For categorical data
- prop.test() – For comparing proportions

Example:

t.test(data$Group1, data$Group2)

This tests whether the mean of two groups is significantly different.

---

50. What is the difference between lm() and glm() in R?

- lm() is used for linear models (continuous output).
- glm() is more general and supports different families such as:
    - **binomial** – for logistic regression
    - **poisson** – for count data

Example:

glm(Survived ~ Age + Fare, data = titanic, family = binomial)

51. How do you interpret the R-squared value in linear regression?

- **R-squared** tells how much variance in the dependent variable is explained by the independent variables.
- Values closer to 1 mean a better fit.
- However, a high R-squared doesn't always indicate a good model – check residual plots too.

52. What is multicollinearity and how do you detect it in R?

Multicollinearity occurs when two or more predictors are highly correlated.

You can detect it using:

- **correlation matrix** – Check if any pair of variables has a correlation > 0.8
- **Variance Inflation Factor (VIF)**:

library(car)

vif(model)

VIF > 5 or 10 usually indicates high multicollinearity.

---

53. How do you perform ANOVA in R?

ANOVA is used to compare means across multiple groups.

```
model <- aov(Sales ~ Region, data = data)
summary(model)
```

- If the p-value < 0.05, there is a significant difference between group means.

---

54. How do you standardize or normalize variables in R?

- **Standardization** (Z-score):

```
data$z_var <- scale(data$var)
```

- **Min-max normalization**:

```
data$norm_var <- (data$var - min(data$var)) / (max(data$var) - min(data$var))
```

Normalization is helpful before applying distance-based algorithms or visualizing data.

---

55. What are confidence intervals and how do you calculate them in R?

A confidence interval gives a range of values likely to contain the true parameter with a given probability (e.g., 95%).

For mean:

```
t.test(data$Income)$conf.int
```

For regression coefficients:

```
confint(model)
```

These intervals help understand the precision of your estimates.

Business Use Cases In R

56. How can R be used for customer segmentation in a business context?

R enables customer segmentation using clustering techniques:

- **K-Means Clustering** is commonly used for segmenting customers by behavior, demographics, or value:

```
set.seed(123)

clusters <- kmeans(scale(data[, c("Age", "SpendingScore")]), centers = 3)

data$Segment <- as.factor(clusters$cluster)
```

- Visualize clusters using ggplot2
- Used in marketing, churn analysis, and personalization strategies

57. How do you perform cohort analysis using R?

Cohort analysis involves grouping users based on their first interaction and tracking behavior over time.

Steps:

- Create a cohort column using the first purchase or signup date
- Calculate retention or engagement metrics per cohort

```
data <- data %>%
  group_by(UserID) %>%
  mutate(Cohort = min(SignupDate)) %>%
  group_by(Cohort, Month) %>%
  summarise(Retention = n())
```

Used in SaaS and e-commerce to track user loyalty.

---

58. How is R used in sales forecasting for businesses?

R supports various forecasting methods such as:

- **Linear regression**
- **Exponential smoothing (ETS)**

- **ARIMA models**

Using forecast package:

```
library(forecast)
model <- auto.arima(sales_ts)
forecast(model, h = 12)
```

Forecasts help in inventory planning and revenue projections.

59. How do you create a sales performance dashboard using R?

Use shiny for interactive dashboards and ggplot2 or plotly for visuals:

- Visualize KPIs: sales, growth %, customer count
- Use filters (date, region, category)
- Update in real-time based on input controls

You can also embed dashboards into HTML or host them using shinyapps.io.

60. How is churn prediction implemented in R for business analysis?

Steps:

1. Prepare labeled data with churn indicators
2. Use classification algorithms like logistic regression or decision trees
3. Evaluate model using confusion matrix, ROC curve

Example:

```
model <- glm(Churn ~ Tenure + Usage + Complaints, family = binomial, data = telecom_data)
summary(model)
```

Used in telecom, SaaS, and e-commerce industries.

61. How do you analyze user behavior over time in R?

Use time-series plots and grouping functions:

```
data %>%
```

```
  group_by(Date, UserID) %>%
  summarise(Sessions = n()) %>%
  ggplot(aes(x = Date, y = Sessions)) +
  geom_line()
```

This helps track DAU (Daily Active Users), WAU (Weekly Active Users), and trends.

---

62. How do you generate automated business reports in R?

Use rmarkdown to create:

- PDF, HTML, or Word reports
- Automatically generated plots, tables, KPIs
- Scheduled reporting scripts

Example RMarkdown structure:

```
---
title: "Monthly Sales Report"
output: pdf_document
---

```{r}
library(ggplot2)
ggplot(sales_data, aes(x = Month, y = Revenue)) + geom_col()
```

yaml
```

---

63. What are the common KPIs that can be calculated using R in business analysis?

Examples:

- Revenue, Profit Margin
- Conversion Rate
- Churn Rate
- Customer Lifetime Value (CLTV)
- Average Order Value
- Customer Acquisition Cost

R can calculate and visualize all these metrics with basic `dplyr` and `ggplot2` operations.

---

64. How do you compare business performance across regions in R?

Use grouping and summarisation:

```R
data %>%
  group_by(Region) %>%
  summarise(AverageSales = mean(Sales), Orders = n())
```

You can visualize it using:

```
ggplot(data, aes(x = Region, y = Sales)) + geom_boxplot()
```

This gives insights into high-performing and underperforming regions.

---

65. How do you conduct A/B testing in R for marketing experiments?

Use hypothesis testing on conversion rates:

```
prop.test(c(Conversions_A, Conversions_B), c(Visitors_A, Visitors_B))
```

Or run a t-test on metric values:

```
t.test(GroupA$CTR, GroupB$CTR)
```

This helps determine if marketing changes led to statistically significant improvements.

Reporting With RMarkdown And Shiny

66. What is RMarkdown and how is it used in reporting?

RMarkdown is a tool in R that allows you to create dynamic reports combining code, output, and narrative text. It supports multiple output formats including HTML, PDF, and Word.

Benefits:

- Reproducible reporting
- Automatically updates plots/tables when data changes
- Integrates R code and text in one document

Example:

markdown

---

title: "Sales Report"

output: html_document

```{r}

summary(sales_data)

yaml

---

---

**67. How do you create a parameterized report in RMarkdown?**

You can define input parameters in the YAML header:

```yaml

params:

region: "East"

Then use them in your R code:

filtered_data <- data %>% filter(Region == params$region)

Useful for generating multiple reports (e.g., one per region or team).

---

**68. What is Shiny and how is it useful for data analysts?**

Shiny is a web application framework for R that allows you to build interactive dashboards and tools using only R code.

Applications:

- Interactive business dashboards
- Filterable reports
- Data input interfaces for clients

You can host Shiny apps locally, on shinyapps.io, or on internal servers.

---

**69. How do you create a basic Shiny app?**

A Shiny app has two core parts:

```
ui <- fluidPage(
  titlePanel("Simple App"),
  sidebarLayout(
    sidebarPanel(sliderInput("num", "Choose a number", 1, 100, 50)),
    mainPanel(textOutput("result"))
)
)

  server <- function(input, output) {
    output$result <- renderText({ paste("You selected", input$num) })
  }
```

```
shinyApp(ui = ui, server = server)
```

This creates a simple interactive application.

---

70. How do you add charts to a Shiny dashboard?

You can embed plots using plotOutput() in the UI and renderPlot() in the server.

Example:

```
ui <- fluidPage(
  plotOutput("sales_plot")
)

 server <- function(input, output) {
   output$sales_plot <- renderPlot({
     ggplot(data, aes(x = Date, y = Sales)) + geom_line()
   })
 }
```

---

71. How do you deploy a Shiny app to the web?

Use shinyapps.io:

1. Create an account on https://www.shinyapps.io
2. Install rsconnect package
3. Deploy using:

```
rsconnect::deployApp("path_to_app")
```

Alternatively, Shiny Server can be used for internal deployments.

---

72. How do you add user input filters to Shiny apps?

Use input controls like selectInput(), checkboxGroupInput(), sliderInput() in the UI and reference their values in the server.

Example:

```
ui <- fluidPage(
```

```
  selectInput("region", "Select Region", choices = unique(data$Region)),
  plotOutput("plot")
)

server <- function(input, output) {
  output$plot <- renderPlot({
    filtered <- data %>% filter(Region == input$region)
    ggplot(filtered, aes(x = Date, y = Sales)) + geom_col()
  })
}
```

---

73. How do you include summary tables in a Shiny dashboard?

Use tableOutput() or renderTable():

```
ui <- fluidPage(
tableOutput("summary")
)

server <- function(input, output) {
  output$summary <- renderTable({
  data %>%
      group_by(Category) %>%
      summarise(Total = sum(Sales))
  })
}
```

For advanced formatting, use DT::renderDataTable().

---

74. How do you combine multiple tabs in a Shiny app?

Use navbarPage() or tabsetPanel():

```
ui <- navbarPage("Dashboard",
  tabPanel("Overview", plotOutput("overview_plot")),
  tabPanel("Details", tableOutput("detail_table"))
)
```

This allows navigation across multiple sections of the dashboard.

---

75. What are best practices when building Shiny dashboards?

- Keep UI clean and minimal
- Limit data loaded in memory for faster performance
- Use reactive() to avoid repeating computations
- Separate UI and server code for larger apps
- Test for responsiveness and cross-browser compatibility

SQL With R

76. How do you connect R to a SQL database?

Use the DBI and RSQLite (or odbc, RMySQL, RPostgreSQL) packages to connect to a database.

Example with SQLite:

```
library(DBI)
con <- dbConnect(RSQLite::SQLite(), "my_database.db")
```

For other databases like PostgreSQL or MySQL, you can use the corresponding drivers and replace the connection string.

77. How do you run SQL queries inside R?

After establishing a connection using DBI, use dbGetQuery() to run SQL queries and return results as a data frame.

Example:

```
result <- dbGetQuery(con, "SELECT * FROM customers WHERE Age > 30")
```

You can also use dplyr::tbl() for a hybrid SQL + R workflow.

78. What is dbWriteTable() used for in R?

dbWriteTable() writes a data frame to a database table.

Example:

```
dbWriteTable(con, "sales_backup", sales_data)
```

It is useful for saving processed data back to a SQL table.

79. How do you parameterize SQL queries in R to avoid SQL injection?

Use placeholders with dbBind() or sqlInterpolate() from DBI or RMySQL.

Example using glue_sql from the glue package:

```
library(glue)
age_limit <- 30
query <- glue_sql("SELECT * FROM users WHERE age > {age_limit}", .con = con)
```

dbGetQuery(con, query)

This is safer and avoids manual string concatenation.

---

80. How can you read large SQL tables in chunks using R?

Use dbSendQuery() and dbFetch():

res <- dbSendQuery(con, "SELECT * FROM large_table")

chunk <- dbFetch(res, n = 1000) # Fetch 1000 rows at a time

Repeat dbFetch() or use a loop to load data in parts for memory efficiency.

81. How can you perform joins using SQL syntax in R?

Simply write SQL join statements within dbGetQuery():

query <- "

SELECT a.CustomerID, a.Sales, b.Region

FROM sales a

JOIN customers b ON a.CustomerID = b.CustomerID"

result <- dbGetQuery(con, query)

This allows complex relational queries to be executed directly from R.

82. What is dbDisconnect() and why is it important?

dbDisconnect(con) is used to close the database connection once all queries are completed.

Not closing the connection can:

- Lead to memory leaks
- Lock the database (especially in SQLite)
- Cause connection limit issues in production

---

83. How can you use dplyr to write SQL queries on remote databases?

You can use the dbplyr package which translates dplyr code into SQL.

Example:

```
library(dplyr)
remote_table <- tbl(con, "sales")
remote_table %>%
  filter(Region == "South") %>%
  summarise(Total = sum(Amount))
```

R will automatically convert this into an optimized SQL query and run it remotely.

---

84. How do you export a SQL query result from R to a CSV file?

Use the following pattern:

```
result <- dbGetQuery(con, "SELECT * FROM orders")
write.csv(result, "orders.csv", row.names = FALSE)
```

This is useful for reporting and offline analysis.

---

85. What are the advantages of using SQL within R for data analysis?

- Combine R's statistical tools with SQL's querying power
- Efficiently handle large datasets via SQL backends
- Simplifies reproducibility for database-driven projects
- Enables secure, controlled data access from production database

---

Time Series And Forecasting In R

86. What is a time series object in R and how do you create one?

A time series object in R is created using the ts() function. It stores data points indexed in time order.

Example:

```
sales_ts <- ts(data$Sales, start = c(2021, 1), frequency = 12)
```

- start defines the time (year, month/quarter)
- frequency = 12 for monthly data, 4 for quarterly, 1 for yearly

---

87. How do you visualize time series data in R?

Use base R or ggplot2:

```
plot(sales_ts)


# Using ggplot2
ggplot(data, aes(x = Date, y = Sales)) +
  geom_line() +
  labs(title = "Monthly Sales Trend")
```

You can also use autoplot() from forecast for ts objects.

---

88. What is decomposition in time series and how is it performed in R?

Decomposition separates a time series into:

- **Trend** – long-term movement
- **Seasonality** – repeating patterns
- **Residuals** – noise

Use decompose() or stl():

```
decomposed <- decompose(sales_ts)
plot(decomposed)
```

This is essential before forecasting to understand data behavior.

---

89. How do you check for stationarity in time series using R?

Use the Augmented Dickey-Fuller (ADF) test from the tseries package:

```
library(tseries)
adf.test(sales_ts)
```

- Null hypothesis: time series is non-stationary
- p-value < 0.05 indicates the series is stationary

Stationarity is required for ARIMA and other forecasting models.

---

90. What is ARIMA and how do you implement it in R?

ARIMA (Auto-Regressive Integrated Moving Average) is a powerful forecasting model.

Use auto.arima() from the forecast package:

```
library(forecast)
model <- auto.arima(sales_ts)
forecasted <- forecast(model, h = 6)
plot(forecasted)
```

- Automatically selects the best (p,d,q) parameters
- Ideal for univariate forecasting

---

91. How do you evaluate a forecasting model in R?

Use these metrics:

- **MAE** (Mean Absolute Error)
- **RMSE** (Root Mean Squared Error)
- **MAPE** (Mean Absolute Percentage Error)

Example:

```
accuracy(forecasted, actual_values)
```

You can also plot actual vs predicted values to visually inspect fit.

---

92. What is the difference between auto.arima() and ets() in R?

- auto.arima() fits ARIMA models (good for data with trends and irregular patterns)
- ets() fits Exponential Smoothing models (good for seasonal data)

Both can be used for forecasting but may perform differently based on the dataset.

---

93. How do you forecast multiple time series in R (e.g., sales by region)?

Use group_by() in combination with models in a loop or via purrr:

```
library(purrr)
models <- data %>%
  group_by(Region) %>%
  group_map(~ auto.arima(ts(.x$Sales, frequency = 12)))
```

You can also use fable and tsibble for a tidy forecasting workflow.

---

94. How do you handle missing values in time series data?

Use na.interp() from the forecast package or imputeTS package:

```
library(forecast)
sales_ts <- na.interp(sales_ts)
```

Other methods:

- na.locf() – Last Observation Carried Forward
- na.approx() – Linear interpolation

---

95. What are the key considerations before building a time series model in R?

Check stationarity – Use ADF test, Visualize seasonality and trends, Handle missing or outlier values, Choose appropriate frequency (monthly, weekly, etc.), Split data into training/testing for evaluation

Proper preprocessing improves forecast accuracy.

Advanced Topics & Optimization In R

96. How do you optimize code performance in R for large datasets?

To make R code more efficient:

- Use data.table for faster data manipulation
- Avoid loops; prefer apply(), lapply(), purrr::map()
- Vectorize operations instead of using for loops
- Use profvis or system.time() to benchmark slow functions
- Process in chunks using readr::read_csv_chunked() or SQL backends

97. How do you parallelize computations in R?

Use parallel, foreach, or future packages:

r

```
library(parallel)
cl <- makeCluster(detectCores() - 1)
clusterExport(cl, "my_function")
parLapply(cl, list_of_inputs, my_function)
stopCluster(cl)
```

Parallel processing helps when dealing with simulations, bootstrapping, or model training on large data.

98. What is lazy evaluation in R and how does it affect function execution?

Lazy evaluation means function arguments are **not immediately evaluated**. They're only evaluated when explicitly used inside the function.

Example:

r

```
fun <- function(x, y) { x }
fun(5, stop("error"))  # No error, because y is not evaluated
```

This can help avoid unnecessary computations, but may cause confusing bugs if not understood well.

---

99. How do you write reusable packages in R?

Use usethis and devtools to create R packages:

r

```
library(usethis)
create_package("myRpackage")
```

Key steps:

- Write functions in /R
- Document using roxygen2